Universidad Carlos III de Madrid

Escuela Politécnica Superior

Posicionador alta-azimutal con controlador de motor I2C basado en ARM7

Proyecto Fin de Carrera
Ingeniería Técnica Industrial: Electrónica Industrial

Autor: María Isabel Lara Puñal
Tutor: Michael García Lorentz

Madrid, Septiembre 2010

El verdadero secreto de la felicidad
consiste en exigirse mucho a sí mismo y
muy poco a los demás.

Albert Guinon

Título: Posicionador alta-azimutal con controlador de motor I2C basado en ARM7
Autor: Maria Isabel Lara Puñal
Director: Michael García Lorentz

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 5 de octubre de 2010 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Tras todos estos años de estudio y vivencias he llegado al final de este camino, aunque en algún momento pensé que no lo lograría. Me gustaría agradecer en esta memoria a todos aquellos que me han acompañado, ya sea desde el principio o sólo durante una etapa.

Gracias principalmente a mis padres por apoyarme siempre ante cualquier bache, por las oportunidades que me han dado, por confiar en mí, por todo, sin ellos no lo hubiese conseguido. Gracias a mi hermano por hacerme entender tantas veces que soy más fuerte de lo que creo. A los miembros de mi familia que han puesto su granito de arena para ayudar a levantarme cuando he caído, mi primo David, mis tías, Natalia.

Quiero dar las gracias a Carlos, Sara, Ana, Álvaro, Fran, Villa, Alberto, Serna, Andrea y a algunos más que olvido mencionar, por las risas, las cañas, los bocatas, la cafetería, las prácticas y todos esos momentos que hemos compartido a lo largo de estos años y que han hecho los ratos de estudio y clases mucho más llevaderos, gracias chicos, habéis hecho de ésta, una etapa maravillosa.

Por supuesto a mis niñas, Sarita, Patri y Elma por la convivencia en la resy y en el piso, por las charlas, el apoyo y el cariño… y por haber secado mis lágrimas en más de una ocasión.

A mis compañeros del BEST por llenar mi vida de experiencias inolvidables, gracias a ellos he aprendido muchas cosas.

Y a Fabián, por aguantar, por apoyarme, por todo… aunque no haya estado al final. También a sus padres.

Finalmente he de agradecer a mi tutor, Michael, la paciencia y el tiempo que me ha dedicado, y a mis profesores, al menos a los que han intentado dar lo mejor de sí mismos en las clases, gracias por todas las materias que habéis hecho que comprenda y aprenda.

La vida está llena de idas y venidas, de personas que entran y salen de tu historia, por eso, a todas aquellas personas que olvido mencionar, gracias por compartir conmigo parte del camino.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

El bus de comunicaciones I2C fabricado por Philips es un puerto estándar altamente utilizado debido a que tan sólo con tres líneas podemos controlar hasta 127 dispositivos.

Entre otras de sus muchas ventajas también se encuentra el hecho de que cada dispositivo es reconocido mediante su dirección y puede operar tanto de transmisor como de receptor en una comunicación bidireccional.

Además el bus permite también el conexionado de varios maestros puesto que incluye un detector de colisiones.

Tanto Philips como otros fabricantes de dispositivos compatibles con I^2C disponen de una amplia gama de circuitos integrados, incluyendo memorias RAM y EEPROM, microcontroladores, puertos de E/S, codificadores DTMF, transceptores IR, conversores A/D y D/A, relojes de tiempo real, calendarios, etc.

Dado que no siempre se requiere alta velocidad de transferencia de datos este bus es ideal para sistemas donde es necesario manejar información entre muchos dispositivos y, al mismo tiempo, se requiere poco espacio y líneas de circuito impreso. Por ello es común ver dispositivos I^2C en video grabadoras, sistemas de seguridad, electrónica automotriz, televisores, equipos de sonido y muchas otras aplicaciones más.

Incluso, y gracias a que el protocolo es lo suficientemente simple, usualmente se ven dispositivos I2C insertados en sistemas microcontrolados que no fueron diseñados con puertos I2C, siendo el protocolo es generado por el firmware.

También hay dispositivos de adaptación que permiten conectar buses originalmente paralelos a sistemas I2C. Tal es el caso del chip PCD 8584 de Philips el cual incorpora bajo su encapsulado todo lo necesario para efectuar dicha tarea.

Hay, además, circuitos integrados cuya única misión es adaptar los niveles presentes en el bus I2C y convertirlos desde y hacia TTL, permitiendo resolver fácil y rápidamente la interconexión de dispositivos de dicha familia con el I2C.

Por lo tanto es un acierto hacer el esfuerzo de tratar de comprender y conectar nuestros dipositivos a través de este protocolo tan extendido.

1.2 Objetivos

El objetivo principal de este proyecto fin de carrera ha sido el control de un montaje alta-azimutal para una cámara de detección de caras al cual se ha dotado de un sensor de temperatura para la detección del sujeto vivo.

Para la consecución de este objetivo principal se ha procedido al desarrollo de un prototipo (software y hardware) usando un microprocesador ARM7, un transductor de temperatura Infra-rojo y un controlador para los motores DC paso a paso que moverán la cámara a la posición deseada.

El prototipo implementado deberá detectar las temperaturas tanto del ambiente como del objeto para poder distinguir posteriormente si éste es un sujeto vivo.

El dispositivo a realizar debe ser capaz de leer una posición enviada por el microprocesador ARM7 y llevar el motor paso a paso a la misma con una velocidad y aceleración determinadas. Al llegar a esa posición el sensor de temperatura actuará para indicar si en ella se encuentra un objeto vivo o por el contrario no.

Como se entiende del modo de funcionamiento expuesto anteriormente, uno de los elementos fundamentales del proyecto será el microprocesador ARM7 y el modo de comunicación entre los diferentes elementos del sistema. Esta comunicación se realizará a través del puerto serie I2C desarrollado por Philips y que éste microprocesador incluye.

Este sistema podrá ser utilizado posteriormente a su optimización por ejemplo en:

- Sistemas de identificación, sobre todo si se combina con un software de reconocimiento de caras.
- Sistemas de seguridad, puesto que la cámara puede ser dotada de numerosas posiciones y en cada una de ellas puede detectar si hay un elemento vivo.
- Sistemas de detección de elementos vivos. Esto puede resultar útil para espacios potencialmente peligrosos para la vida como aquellos que van a ser utilizado para pruebas químicas o armamentísticas.
- Sistemas de seguimiento en los cuales la cámara pueda ir moviéndose hacia las posiciones en las cuales se vaya detectando temperatura de elementos vivos en movimiento.

1.3 Fases de desarrollo

Para el desarrollo del dispositivo prototipo se han realizado diversas etapas que se detallan a continuación.

1.3.1 Estudio previo

Como se ha comentado con anterioridad, ha sido necesario un estudio previo exhaustivo tanto de la tecnología empleada como de su configuración.

Se comenzó con el estudio del microprocesador ARM7 puesto que iba a ser utilizado como maestro del sistema. Se estudió tanto su constitución física como sus memorias, registros y modos de utilización.

Además, puesto que utilizaríamos la placa de pruebas MCB2100 se hizo necesario también un estudio de las conexiones de la misma y de sus medidas.

Posteriormente se estudió la tecnología I2C, su funcionamiento, el protocolo de comunicación a seguir, los registros a utilizar, el modo de transmisión de los datos y su orden etc. La comprensión de ésta tecnología es fundamental para el desarrollo del proyecto por lo tanto hicimos mucho hincapié en su asimilación.

Una vez comprendido el modo de comunicación y el funcionamiento del dispositivo que iba a ser el maestro se procedió al estudio de elementos esclavos del sistema, es decir, el sensor y el motor.

Primeramente se estudió el sensor tanto a nivel hardware como a nivel software. Se estudiaron sus conexiones y el modo de alimentarlo, así se vio la necesidad de utilizar unas resistencias pull-up y condensadores de desacoplo. Después se procedió al estudio de las memorias internas y registros a utilizar para su configuración así como su configuración previa; el formato de los datos, el proceso de lectura de la temperatura etc.

En segundo lugar se estudiaron conjuntamente el controlador utilizado y el motor que controlaría. El proceso de estudio fue similar al del sensor, primero se estudió a nivel físico y después se estudiaron sus memorias internas y todos los comandos necesarios para el adecuado movimiento del motor.

Tras la comprensión de todos los elementos de nuestro prototipo y su modo de comunicación se estudiaron también las herramientas que se utilizarían para su configuración, es decir, los programas Keil micro visión 3 y el Orcad.

1.3.2 Desarrollo software

Para desarrollar el software del dispositivo primeramente decidimos la estructura que seguiría el código en función de los objetivos perseguidos, es decir, qué debíamos hacer y cómo lo haríamos.

Comenzamos decidiendo que realizaríamos la configuración de los dispositivos con los datos obtenidos de las hojas de fabricante esto fue, para el caso del sensor, las temperaturas y para el caso del motor, las posiciones velocidades y aceleraciones.

Puesto que no contábamos con una posición específica y una temperatura específica ya que esto variará dependiendo de la aplicación del dispositivo se procedió a la configuración de los elementos únicamente.

Se decidió estructurar el software de manera que fuese muy sencillo variar los datos a enviar y recibir en caso de querer cambiar la aplicación o los valores predeterminados del sistema, por lo tanto desarrollamos el software como un conjunto de buffers de escritura y lectura fácilmente modificables. Estos buffers son enviados a las distintas funciones que realizarán la configuración de los dispositivos.

Esto es muy positivo puesto que el código contenedor de las funciones, que es el encargado de llevar a cabo la comunicación siguiendo el protocolo I2C, no ha de modificarse en absoluto en caso de querer cambiar los datos a enviar o recibir.

Tanto para el sensor como para el motor se siguió la misma estructura que se detallará más adelante en esta misma memoria.

1.3.3 Desarrollo hardware

El esquema hardware fue sencillo de implementar tras el estudio previo del conexionado de todos los dispositivos.

Las únicas complicaciones surgieron ante la inexistencia en el programa Orcad Layout de librerías que pudiesen utilizarse para nuestros componentes, es decir, para el sensor, las resistencias pull-up y el controlador.

Para ello tuvimos que realizar nuestras propias librerías con las medidas indicadas en las hojas de fabricantes.

Además tuvimos que tener en cuenta que la placa debía encajar en la MCB2100 que utilizaríamos.

Por lo tanto el desarrollo del hardware posterior al estudio consistió primeramente, en la realización de las librerías para los componentes, en segundo lugar se procedió a la realización del esquema de conexiones siguiendo las indicaciones de las diferentes hojas de características y finalmente se realizó el diseño del rutado de la placa.

Esta placa debía ser implementada en el laboratorio en el cual se realizarían las pistas diseñadas para la posterior colocación de los elementos mediante soldadura.

Tras tener ésta placa con todos sus elementos, se conectaría a la placa de pruebas MCB2100 y al motor para realizar la configuración del software de los mismos. Para esta configuración la placa MCB2100 incluye un conector que permite descargar el software implementado en nuestro ordenador con la herramienta Keil, directamente a la placa.

1.4 Esquema de la memoria

El esquema que se ha seguido para la elaboración de esta memoria está en concordancia con el desarrollo de las fases del mismo.

Capitulo 1:

En él se incluye la introducción, la descripción de objetivos y las fases de desarrollo del proyecto.

Capítulo 2:

Explicación y descripción del estado del arte de la cuestión, de los elementos utilizados en nuestro prototipo y del estudio previo de los mismos realizado.

Capitulo 3:

Capítulo destinado a la descripción detallada del prototipo realizado. Se comenta primeramente el software implementado, haciendo hincapié en los datos enviados para la configuración de los dispositivos, y posteriormente se describe el diseño hardware con el conexionado correspondiente.

Capitulo 4:

Presentación de las medidas y pruebas realizadas. Dichas medidas son simulaciones del funcionamiento correcto del código, es decir, del envío y recepción de los mensajes entre los diferentes dispositivos.

Capítulo 5:

Destinado a la conclusión, las líneas futuras de desarrollo del proyecto y el presupuesto.

Capítulo 2

Estado del arte

En este capítulo se van a detallar primeramente algunos aspectos importantes sobre los elementos y la tecnología empleada en el desarrollo del prototipo como son los motores paso a paso, las resistencias pull-up, los transductores de temperatura y los microprocesadores. Posteriormente explicaremos el funcionamiento de la tecnología I2C de comunicación.

2.1 Motores Paso a paso

Utilizaremos un motor de tipo paso a paso para el movimiento de la cámara en los ejes alfa-azimutal.

Los motores en general son máquinas eléctricas rotativas capaces de transformar energía eléctrica en energía mecánica, y los motores Paso a paso (*PaP*) son un caso muy particular. En ellos la alimentación no es ni corriente alterna ni corriente continua sino un tren de pulsos.

Esta es la característica principal de los motores paso a paso, el poder moverlos un paso a la vez por cada pulso que se le aplique.

El tren de pulsos se sucede con una secuencia, previamente definida, a cada una de las bobinas que componen el estator (*Los motores estarán compuestos por un rotor y un estator que posteriormente se describen)*. Dependiendo de las características constructivas del motor este paso puede ser desde 90º hasta incluso 0,9º.

Por lo tanto, si somos capaces de mover el motor en pequeños pasos, esto nos va a permitir controlar su posición, con mayor o menor precisión dependiendo del avance de cada paso.

Además, variando la frecuencia con la que se aplican los pulsos, también estaremos variando la velocidad con que se mueve el motor, lo que nos permite realizar un control de velocidad.

Por último si invertimos la secuencia de los pulsos de alimentación aplicados a las bobinas, estaremos realizando una inversión en el sentido de giro del motor.

Estos motores poseen la habilidad de poder quedar enclavados en una posición o bien totalmente libres. Si una o más de sus bobinas esta energizada, el motor estará enclavado en la posición correspondiente y por el contrario quedará completamente libre si no circula corriente por ninguna de sus bobinas. Esta característica se denomina "par de detención", e incluso par/torque "de mantenimiento" y no existe en los motores de CC. El torque de detención hace que un motor paso a paso se mantenga firmemente en su

posición cuando no está girando. Se elimina así la necesidad de un mecanismo de freno.

Entre las principales aplicaciones de estos motores se pueden mencionar la robótica en general, la tecnología aeroespacial, el control de discos duros, de discos flexibles (disquetes), unidades de CD-ROM o de DVD e impresoras de todo tipo.

Si cabe citar algún inconveniente de los motores PaP sería que presentan una velocidad angular limitada. Dicha limitación surge porque para realizar un paso, el motor requiere un tiempo para alcanzar la posición de equilibrio. Si dicho tiempo no se respeta (esto ocurriría si la frecuencia de los pulsos es demasiado elevada) el motor puede no encontrar nunca esa posición de equilibrio y perderíamos el control sobre él (se mueve en forma de vaivén, no se mueve, o incluso se mueve en sentido contrario al deseado).

Figura 1: Motor Paso a paso

2.1.1 Tipos de motores PaP

Desde el punto de vista constructivo existen tres tipos de motores PaP [web1]:

De imán permanente:

Es el tipo de motores PaP más utilizado, y sus características constructivas son las siguientes:

- El **rotor** está formado por un imán permanente, en forma de disco, y en cuya superficie se encuentran mecanizados un determinado número de dientes**.**

- El **estator** tienen forma cilíndrica, y en su interior se encuentran diversos bobinados, que al ser alimentados secuencialmente generan un campo magnético giratorio.

- Como resultado de las fuerzas de atracción-repulsión, el rotor se orientará dentro de este campo magnético giratorio, lo que provocará su movimiento controlado.

- La conmutación en la alimentación de las bobinas tiene que ser manejada por un controlador externamente.

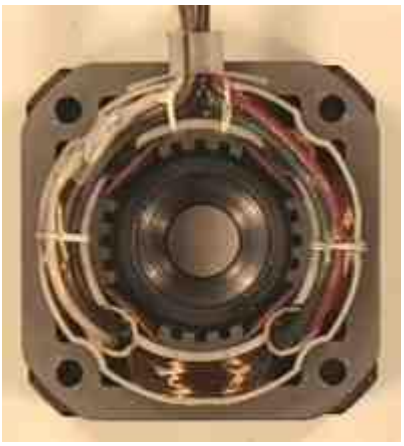


Figura 2: Estator de un motor PaP de imán permanente

Figura 3: Rotor de un motor PaP de imán permanente

De reluctancia variable:

- El **estator** es similar al caso anterior.

- El **rotor** no es un imán permanente, sino que está formado por un núcleo de hierro dulce, e igualmente con dientes tallados a lo largo de su superficie.

- En este tipo de motor, al alimentar una de las bobinas del estator, se crea un campo magnético. En estas condiciones, el rotor se orienta hacia aquella posición en la que la reluctancia que presenta el circuito es mínima. Esta posición será aquella en la que el entrehierro sea el más

pequeño posible. Al cambiar la alimentación a otra de las bobinas, el punto de mínima reluctancia también cambia, con lo cual el rotor gira de nuevo.

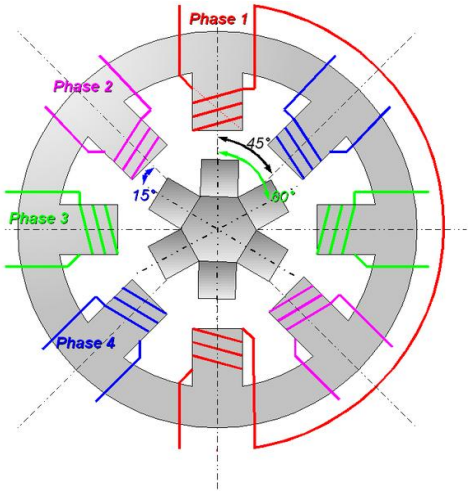


Figura 4: Esquema de un motor PaP de reluctancia variable

Híbridos:

Este tipo de motores son una mezcla de los dos anteriores. El rotor está formado por una serie de anillos de acero dulce que tienen en su superficie un nº de dientes ligeramente distinto a los del estator. Dichos anillos están montados sobre un eje que es un imán permanente.

En nuestro prototipo hemos utilizado motores de imán permanente por lo tanto en adelante nos centraremos en éstos, además, son los más utilizados.

2.1.2 Tipos de motores PaP de imán permanente

Los motores de imán permanente pueden ser clasificados, en función del sentido de la intensidad que recorre los bobinados ,en dos grupos:

Motores PaP bipolares:

Están formados por dos bobinas y la intensidad que circula por ellas invierte su sentido sucesivamente (de ahí surge el nombre de bipolares). Se pueden reconocer externamente porque presentan cuatro conductores, uno para cada extremo de una bobina.

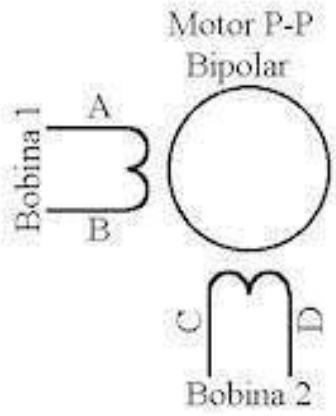


Figura 5: Esquema de un bobinado bipolar

Motores PaP unipolares:

En este caso el estator está formado por dos bobinas con tomas intermedias, lo que equivale a cuatro bobinas. Las tomas intermedias de las dos bobinas pueden estar interconectadas en el interior o no. Externamente se apreciarán cinco conductores en el primer caso, y seis en el segundo. La forma de alimentar este motor consiste en poner a masa la toma central e ir aplicando según una secuencia determinada pulsos de valor +V a un extremo de la bobina y al otro (nunca simultáneamente). De tal manera que la intensidad que circula por cada media bobina siempre lo hace en el mismo sentido, por eso se denominan unipolares. Otra posibilidad de alimentación, consiste en dejar fija en la toma intermedia una tensión +V, e ir alternando en ambos extremos la conexión con masa.

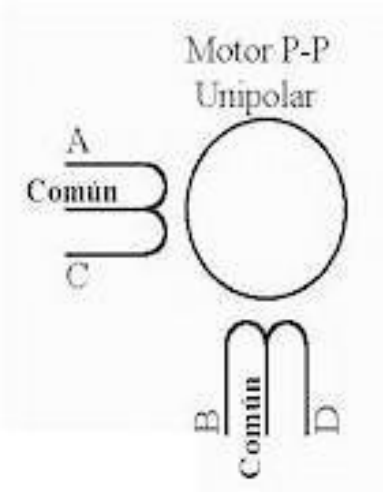


Figura 6: Esquema de un bobinado unipolar

2.1.3 Funcionamiento de PaP de imán permanente

Básicamente, los motores PaP de imán permanente están constituidos normalmente por un rotor sobre el que van aplicados distintos imanes permanentes y por un cierto número de bobinas excitadoras bobinadas en su estator. Las bobinas son parte del estator y el rotor es un imán permanente. Toda la conmutación (o excitación de las bobinas) deber ser externamente manejada por un controlador. Es decir, el principio de funcionamiento de este tipo de motores está basado en las fuerzas de atracción-repulsión que experimentan los cuerpos sometidos a un campo magnético.

Motores PaP bipolares:

Este tipo de motor lleva dos bobinados independientes el uno del otro, y como ya se ha comentado, para controlar este motor se necesita invertir la polaridad de cada una de las bobinas en la secuencia adecuada, para esto necesitaremos usar un puente en "H" para cada bobina del motor, es decir que para controlar un motor Paso a Paso de 4 cables (dos bobinas), necesitaremos usar dos puentes "H" iguales al de la figura 7. En general es recomendable el uso de puentes "H" integrados como son los casos del L293 [web2].

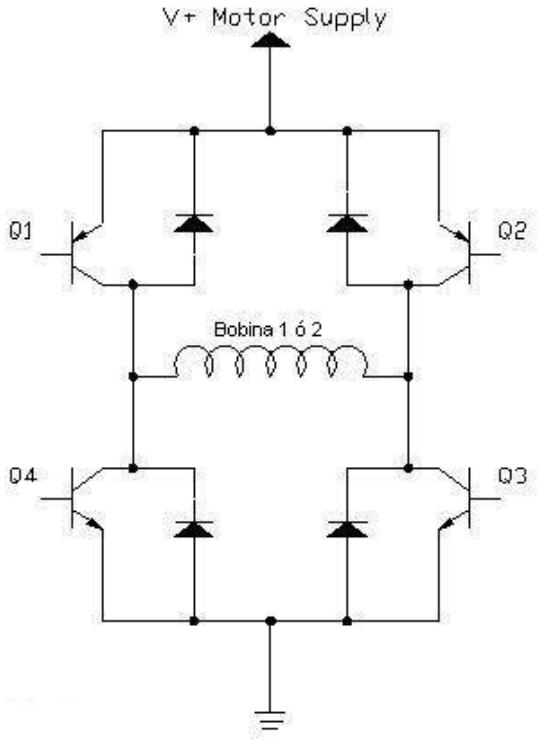


Figura 7. Puente "H" modo ilustrativo

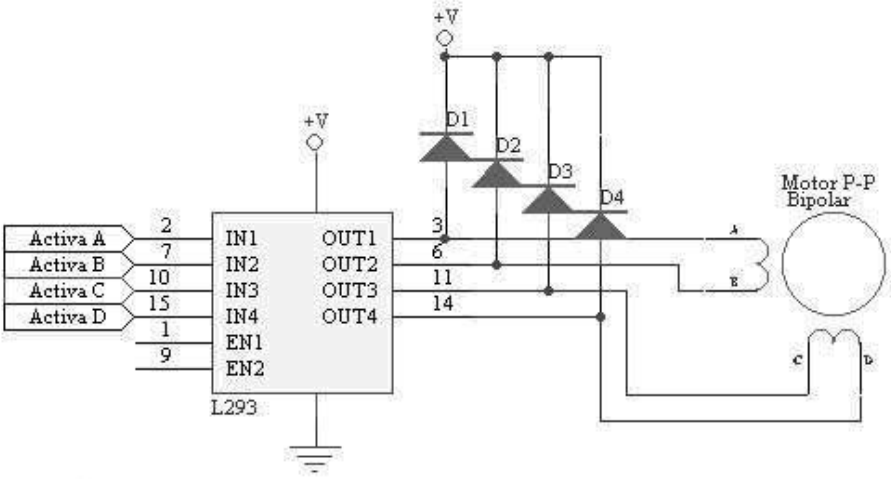


Figura 8: Integrado L293: Puente "H"

Para que el motor gire en hacia la derecha (sentido horario), debemos aplicar a las bobinas intensidad en una secuencia determinada, la cual se detalla en la Tabla 1.

Nº de Pasos	1a	1b	2a	2b
Paso 1	+Vcc	GND	+Vcc	GND
Paso 2	+Vcc	GND	GND	+Vcc
Paso 3	GND	+Vcc	GND	+Vcc
Paso 4	GND	+Vcc	+Vcc	GND

Tabla 1 : Secuencia de giro del motor bipolar

Cada inversión en la polaridad provoca el movimiento del eje, avanzando este un paso, la dirección de giro se corresponde con la dirección de la secuencia de pasos, por ejemplo para avanzar el sentido horario la secuencia seria 1-2-3-4,1-2-3-4.... y para sentido anti-horario seria; 4-3-2-1,-4-3-2-1...[web3]

Motores PaP unipolares

Recordemos que un motor unipolar está compuesto por dos bobinas con una toma intermedia cada una, y su principal característica es que la intensidad que circula por cada bobina siempre lo hace en la misma dirección.

El motor unipolar normalmente dispone de 5 o 6 cables dependiendo si el común esta unido internamente o no, para controlar este tipo de motores existen tres métodos con sus correspondientes secuencias de encendido de bobinas, el común irá conectado a +Vcc o masa según el circuito de control usado y luego tan solo tendremos que alimentar la bobina correcta para que

avance o retroceda el motor según avancemos o retrocedamos en la secuencia, para invertir el sentido de giro tan sólo habrá que ejecutar las secuencias inversamente.

Las secuencias son las siguientes [web4]:

- **Paso simple:**

Esta secuencia de pasos es la mas simple de todas y consiste en activar cada bobina una a una y por separado, con esta secuencia de encendido de bobinas no se obtiene mucha fuerza ya que solo es una bobina cada vez la que arrastra y sujeta el rotor del eje del motor (ver Figura 9)

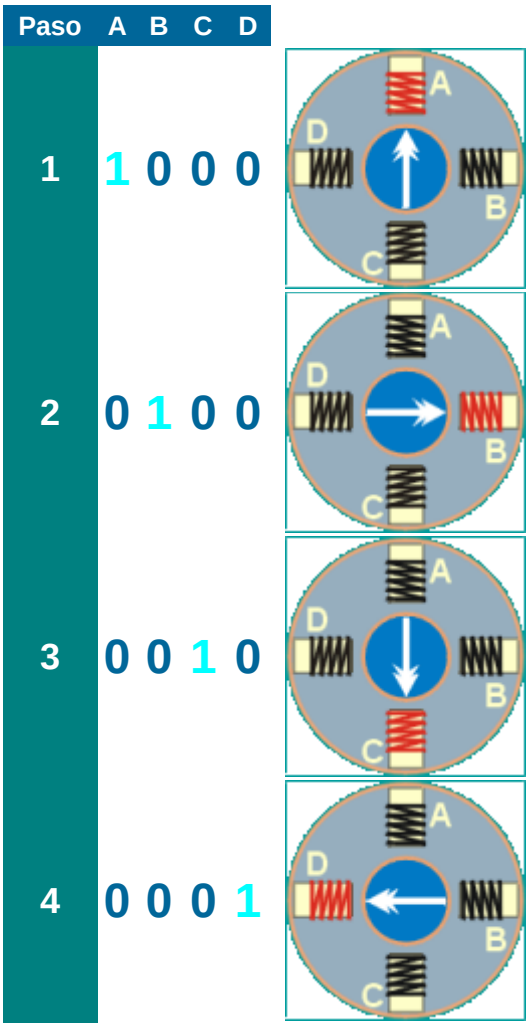


Figura 9: Secuencia paso simple en motor PaP unipolar

- **Paso doble:**

Con el paso doble activamos las bobinas de dos en dos con lo que hacemos un campo magnético más potente que atraerá con más fuerza y retendrá el rotor

del motor en el sitio. Los pasos también serán algo mas bruscos debidos a que la acción del campo magnético es mas poderosa que en la secuencia anterior.

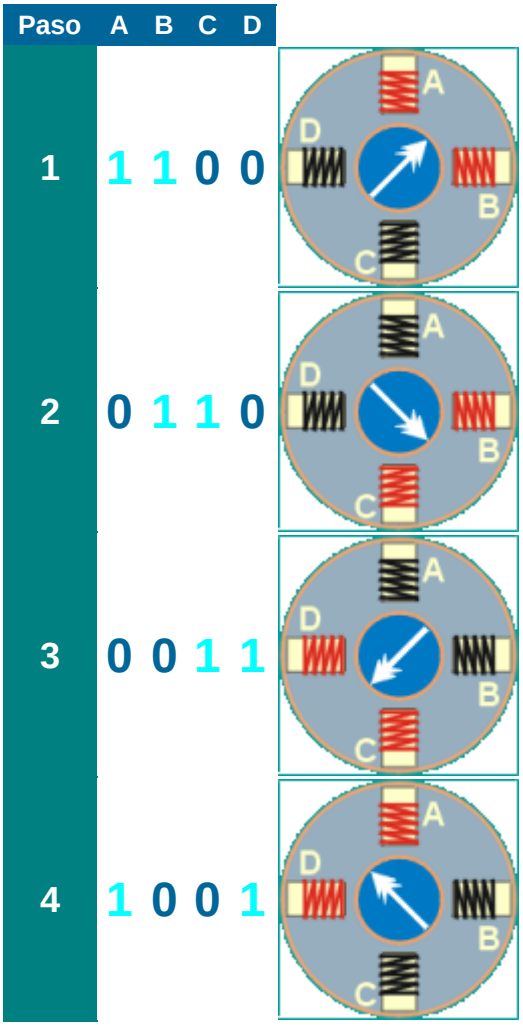
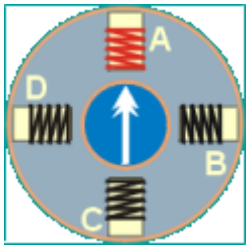
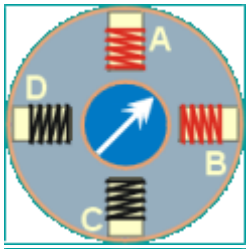
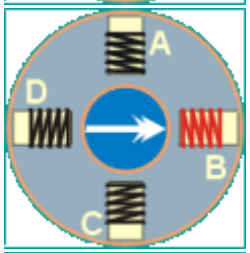
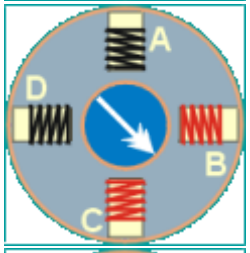
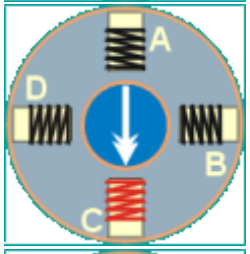
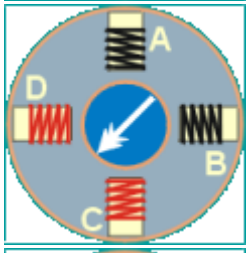
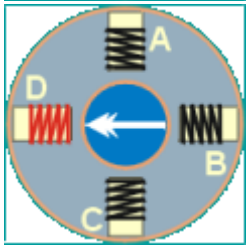


Figura 10: Secuencia paso doble en motor PaP unipolar

- **Medio Paso:**

Combinando los dos tipos de secuencias anteriores podemos hacer moverse al motor en pasos más pequeños y precisos y así pues tenemos el doble de pasos de movimiento para el recorrido total de 360º del motor. En esta secuencia se activan las bobinas de tal forma que brindan un movimiento igual a la mitad del paso real. Para ello se activan primero 2 bobinas y luego solo 1 y así sucesivamente.

Paso	A	B	C	D	
1	1	0	0	0	
2	1	1	0	0	
3	0	1	0	0	
4	0	1	1	0	
5	0	0	1	0	
6	0	0	1	1	
7	0	0	0	1	

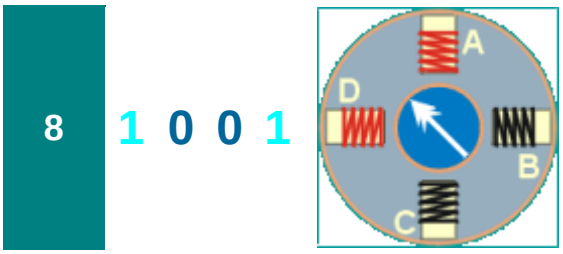


Figura 11: Secuencia paso medio en motor PaP unipolar

Como comentario final, cabe destacar que debido a que los motores paso a paso son dispositivos mecánicos y como tal deben vencer ciertas inercias, el tiempo de duración y la frecuencia de los pulsos aplicados es un punto muy importante a tener en cuenta. En tal sentido el motor debe alcanzar el paso antes que la próxima secuencia de pulsos comience. Si la frecuencia de pulsos es muy elevada, el motor puede reaccionar en alguna de las siguientes formas:

- Puede que no realice ningún movimiento en absoluto.
- Puede comenzar a vibrar pero sin llegar a girar.
- Puede girar erráticamente.
- O puede llegar a girar en sentido opuesto.

Para obtener un arranque suave y preciso, es recomendable comenzar con una frecuencia de pulso baja y gradualmente ir aumentándola hasta la velocidad deseada sin superar la máxima tolerada. El giro en reversa debería también ser realizado previamente bajando la velocidad de giro y luego cambiar el sentido de rotación.

2.1.4 Motor utilizado: QSH4218

Se pueden utilizar multitud de motores para desarrollar el prototipo dependiendo de las diversas aplicaciones.

El motor propuesto es un motor de PaP híbrido de dos fases optimizado para pasos muy pequeños y que se han desarrollado para ser utilizados junto a los controladores de la misma familia (Trinamic) (ver figura 1).

Las características principales de esos motores son:

- Angulo de paso 1’8º
- Voltaje de operación hasta 42v
- Conexionado de 4 cables
- Imanes de neodimio
- 42’2mm x 42’2mm

2.1.5 Controlador del motor utilizado: TMC222

Como se ha comentado anteriormente, es necesario un controlador para utilizar los motores PaP.

En nuestro prototipo se ha utilizado este controlador puesto es compatible con todos los tipos de motor y con la interfaz I2C que utilizaremos para la comunicación tanto con el motor como con el sensor de temperatura.

El TMC222 es un dispositivo que combina el controlador de motores PaP y los drivers junto con una memoria RAM y otra memoria OTP utilizadas para configurar el motor y almacenar los datos necesarios para el correcto movimiento del motor.

Las características principales de este controlador son:

- Velocidad y aceleración configurables
- Posicionamiento autónomo
- Control de velocidad
- Memorias programables
- Utilizable a altas temperaturas

Figura 12: Controlador TMC222

2.2 Transductores de temperatura

La temperatura es una de las magnitudes físicas que se miden con mayor frecuencia, pero es un factor de medida engañoso debido a su simplicidad. A menudo pensamos en ella como un simple número, pero en realidad es una estructura estadística cuya exactitud y repetitividad pueden verse afectadas por la masa térmica, el tiempo de medida, el ruido eléctrico y los algoritmos de medida.

2.2.1 Qué es la temperatura

La temperatura está definida como la energía promedio por molécula de un material [web5].

La temperatura se mide básicamente a partir de cambios en las propiedades de diversos materiales al ser influidos por la temperatura entre los cuales podemos contar:

- Variaciones en el volumen o en el estado de los cuerpos (sólidos, líquidos o gases).
- Variaciones en la resistencia de algún conductor (sondas de resistencias).
- Variación en la resistencia de algún semiconductor (termistores).
- Fuerza electromotriz generada en la unión de dos metales distintos (termopares).
- Intensidad de la radiación emitida por un cuerpo (pirómetros de radiación).
- Fenómenos utilizados en el laboratorio (velocidad del sonido de un gas, frecuencia de resonancia de un cristal, etc.).

En nuestro prototipo utilizaremos un termistor Infra-rojo que entraría en la categoría de pirómetros de radiación.

Ningún transductor es el mejor en todas las situaciones de medida, por lo que tenemos que saber cuándo debe utilizarse cada uno de ellos. En la Tabla 2 se están comparando los cuatro tipos de transductores de temperatura más utilizados reflejando los factores que deben tenerse en cuenta: las prestaciones, el alcance efectivo, el precio y la comodidad [web6].

	RTD	**Termistor**	**Sensor de IC**	**Termopar**
Ventajas	Más estable. Más preciso. Más lineal que los Termopares.	Alto rendimiento Rápido Medida de dos hilos	El más lineal El de más alto rendimiento Económico	Autoalimentado Robusto Económico Amplia variedad de formas físicas Amplia gama de temperaturas
Desventajas	Caro. Lento. Precisa fuente de alimentación. Pequeño cambio de resistencia. Medida de 4 hilos Autocalentable	No lineal. Rango de Temperaturas limitado. Frágil. Precisa fuente de alimentación. Autocalentable	Limitado a < 250 ºC Precisa fuente de alimentación Lento Autocalentable Configuraciones limitadas	No lineal Baja tensión Precisa referencia El menos estable El menos sensible

Tabla 2 :Comparativa de termistores

2.2.2 Tipos de transductores de temperatura

En este apartado estudiaremos los tipos de transductores más utilizados y posteriormente describiremos detalladamente el tipo de transductor empleado en el prototipo diseñado.

Detectores de temperatura resistivos (RTD)

Los detectores de temperatura resistivos (RTD; **R**esistive **T**emperature **D**etector) son los sensores de temperatura más estables y precisos [web7].

El material de fabricación hace variar el rango de medida pero no de forma apreciable. En consecuencia, estos transductores se emplean en aplicaciones que requieren alta precisión y repetibilidad, como control de calidad de alimentos y aplicaciones farmacéuticas. La precisión suele expresarse como un porcentaje de la resistencia nominal a una temperatura dada.

El elemento consiste usualmente en un arrollamiento de hilo muy fino del conductor adecuado bobinado entre capas de material aislante y protegido con un revestimiento de vidrio o de cerámica.

Los materiales más comunes para la fabricación de RTD's son el platino, el níquel y el cobre y en algunas aplicaciones a bajas temperaturas (alrededor de los 20^{o}K) el Rodio.

Las características que deben tener los materiales empleados son:

- Alto coeficiente de temperatura de la resistencia (sensibilidad).
- Alta resistividad, ya que cuanto mayor sea la temperatura mayor será la variación por grado.
- Relación lineal temperatura-resistencia.
- Rigidez y ductilidad.
- Estabilidad de las características en la vida útil.

La relación entre factores puede verse en la expresión lineal siguiente [web8]:

$$\mathbf{R_t = R_0 (1 + \alpha t)}$$

En la que:

$\mathbf{R_0}$ = Resistencia en ohmios a 0°C.
$\mathbf{R_t}$ = Resistencia en ohmios t °C.
α = Coeficiente de temperatura de la resistencia.

El llamado ***"coeficiente de temperatura de resistencia"*** expresa, a una temperatura especificada, la variación de la resistencia en ohmios del conductor por cada grado que cambia su temperatura.

Una desventaja del uso de RTD's es que, para medir la resistencia hay que aplicar una corriente que, por supuesto, produce una cantidad de calor que distorsiona los resultados de la medid, por lo tanto, se deben emplear corrientes que circulen a través de ellos lo suficientemente pequeñas para evitar el autocalentamiento.

Otra desventaja es que el material más fiable es el platino, que permite realizar medidas más exactas y estables hasta una temperatura de aproximadamente 500 ºC y este material encarece mucho el dispositivo. Los RTD más económicos utilizan níquel o aleaciones de níquel, pero no son tan estables ni lineales como los que emplean platino.

Una tercera desventaja, que afecta al uso de este dispositivo para medir la temperatura, es la resistencia de los RTD. Al ser tan baja, la resistencia de los hilos conductores que conectan el RTD puede provocar errores importantes. Esto se puede subsanar utilizando métodos de medida como las técnicas de los dos, cuatro o tres hilos (figura 13), con los cuales se intenta que la resistencia de la RTD no afecte a la medida [web9].

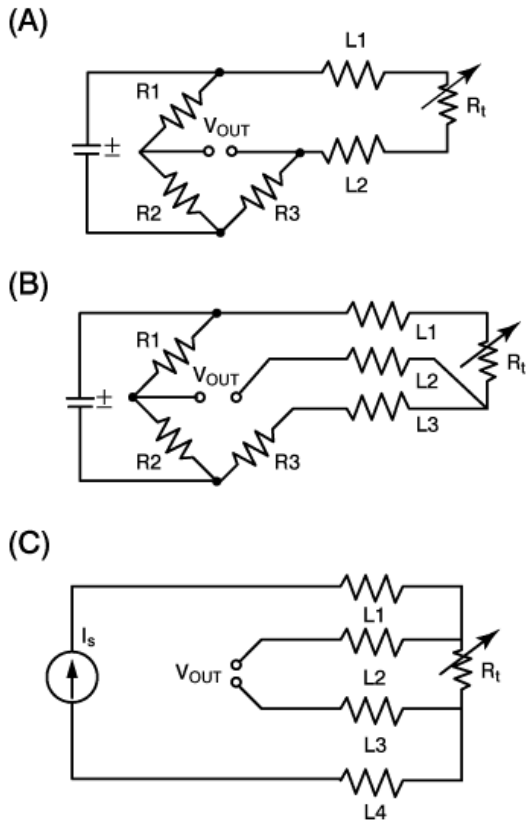


Figura 13: Métodos de medida de la temperatura con termistores RTD.(A)método de los dos hilos, (B)método de los tres hilos, (C)método de los cuatro hilos.

Termistores

Un termistor es un sensor resistivo de temperatura. Su funcionamiento se basa en la variación de la resistividad que presenta un semiconductor con la temperatura, debido a la variación de la concentración de portadores. Existen dos tipos de termistor:

- NTC (Negative Temperature Coefficient) – coeficiente de temperatura negativo
- PTC (Positive Temperature Coefficient) – coeficiente de temperatura positivo

Figura 14: Termistor PTC

Para los termistores NTC, al aumentar la temperatura, aumentará también la concentración de portadores, por lo que la resistencia será menor, de ahí que el coeficiente sea negativo. Para los termistores PTC, en el caso de un semiconductor con un dopado muy intenso, éste adquirirá propiedades metálicas, tomando un coeficiente positivo en un margen de temperatura limitado.

Usualmente, los termistores se fabrican a partir de óxidos semiconductores, pueden ser mezclas sintetizadas de sulfatos, selenio, óxidos de níquel, manganeso, hierro, cobalto, cobre, magnesio, titanio, uranio, y otros metales, y para las PTC, normalmente se utilizan fabricados de bario sintetizado y mezclas de estroncio y titanio.

A diferencia de los sensores RTD, la variación de la resistencia con la temperatura es no lineal. En consecuencia, a diferencia de los RTD que son de propósito general, se emplean en aplicaciones que requieren medidas sensibles sobre un rango pequeño de temperatura.

La ecuación que domina el cambio de resistencia de un termistor respecto a la temperatura está dada por:

$$R_T = R_0 e^{b\,(1/Tt\,-1/To)}$$

Donde:

R_T = Resistencia del termistor.
R_0 = Resistencia inicial.
b = Coeficiente térmico.
Tt = Temperatura de trabajo en Kelvin.
To = Temperatura de referencia.

Por tratarse de material semiconductor, los termistores tienen un rango limitado que va de -20ºC a 150ºC, y como se puede apreciar en la ecuación anterior, su respuesta es no lineal por el término exponencial, además son dispositivos que presentan el fenómeno de envejecimiento. Las curvas de cambio de resistencia en función de la temperatura para termistores con NTC y PTC se muestran en la figura 15.

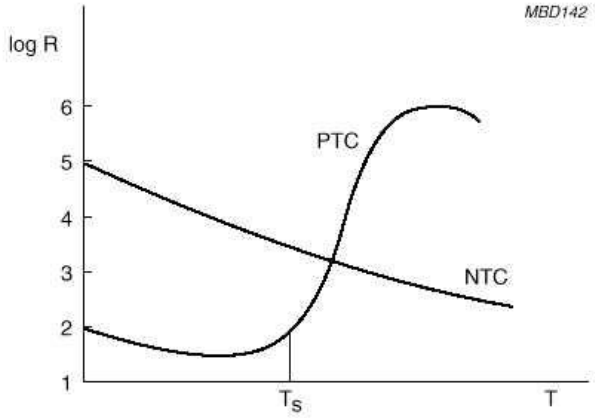


Figura 15: Variación Resistencia-Temperatura en termistores

Como características más ventajosas a la hora de utilizar termistores podemos destacar que, dada la alta resistividad de los materiales empleados, es posible disponer de termistores de tamaño reducido. Esto, por un lado, permite reducir el coste del sensor y su empleo en una amplia variedad de aplicaciones y también permite reducir bastante el tiempo de reacción del sensor [web10].

Por otro lado los termistores son más susceptibles al autocalentamiento que los sensores RTD, debido a su pequeño tamaño, lo cual, es un inconveniente.

Sin embargo, la sensibilidad de un termistor puede ser bastante elevada, con grandes variaciones de resistencia ante pequeñas variaciones de temperatura. Esto facilita su utilización, ya que el error introducido por los hilos de interconexión puede considerarse despreciable.

Debemos tener en cuenta a pesar de esto, que existen una serie de desventajas, como por ejemplo, que el margen de funcionamiento de un termistor, de -100 °C a 450 °C, es más limitado que en los termopares y los sensores RTD.

Otra desventaja notable es que la variación de la resistencia con la temperatura para un termistor, comparada con un sensor RTD, es muy no lineal, sin embargo existen diversos modos de linealización de la media a costa de añadir más componentes al circuito como resistencias en paralelo.

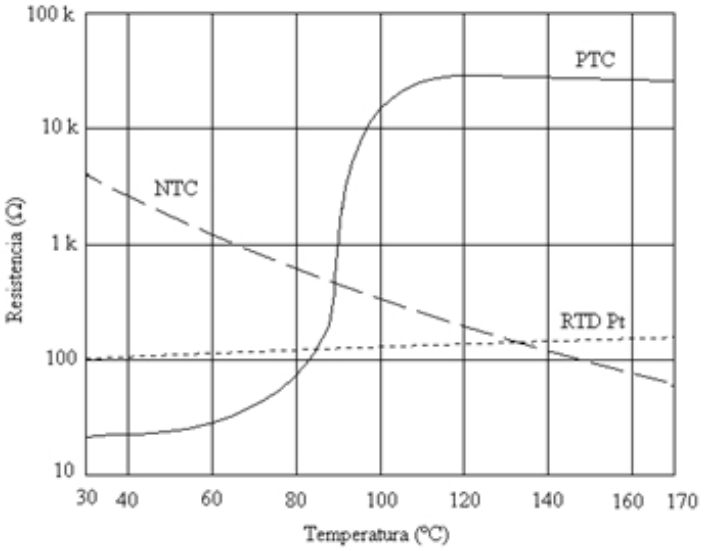


Figura 16: Comparativa termistores-RTD en cuanto a linealidad

Una segunda desventaja sería que para los termistores no hay disponibles curvas de calibración estándar, por lo que intercambiar sensores entre distintos fabricantes puede suponer un problema, llegando incluso a la necesidad de rediseñar parte del sistema. Cada fabricante posee sus propias curvas de variación de la resistencia con la temperatura.

En definitiva, los termistores son los más apropiados para aplicaciones en las que sean necesarias medidas rápidas de temperatura con suficiente sensibilidad. Su pequeño tamaño puede resultar beneficioso cuando el espacio es crítico, teniendo en cuenta siempre los errores debido a autocalentamiento.

Termopares

Los termopares son probablemente los sensores de temperatura más extendidos tanto en la industria como en los laboratorios. Se emplean sobre todo en situaciones de adquisición masiva de datos, pero, a pesar de lo extendido de su empleo, no es sencillo lograr el correcto manejo del transductor, debido a que existen muchos tipos de termopares y sus datos requieren tratamiento para obtener resultados válidos.

Los termopares son elementos de medición de temperatura activos cuyo principio operativo es el llamado efecto *Seebeck* que Thomas Seebeck descubrió en 1812. Esto es, que la unión de dos metales distintos genera un potencial (una corriente continua si el circuito es cerrado) en función de la temperatura a la que se somete la unión, es decir, estos sensores no miden la temperatura absoluta, sino que realizan una comparativa con una temperatura de referencia.

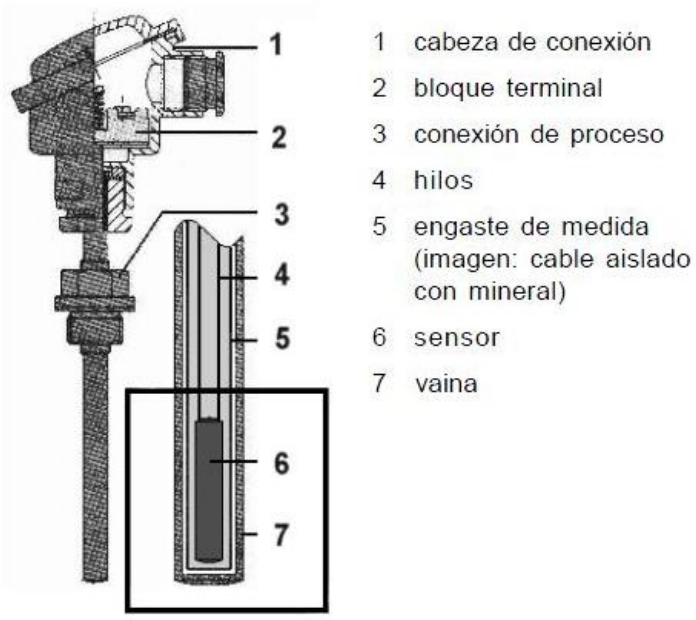


Figura 17: Esquema de sonda con termopar

El termopar se compone de dos conductores de diferente coeficiente térmico unidos en sus extremos.

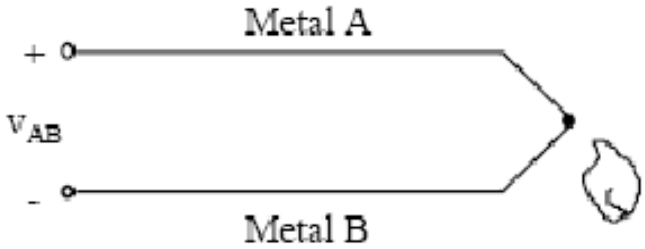


Figura 18: Esquema de un termopar

Estudios realizados sobre el comportamiento de termopares han permitido establecer tres leyes fundamentales:

I. *Ley del circuito homogéneo*. En un conductor metálico homogéneo no puede sostenerse la circulación de una corriente eléctrica por la aplicación exclusiva de calor. Ahora, si dos metales, A y B, son sometidos a temperaturas T1 y T2 en sus uniones entonces, existirá una fuerza electromotriz generada en los extremos.

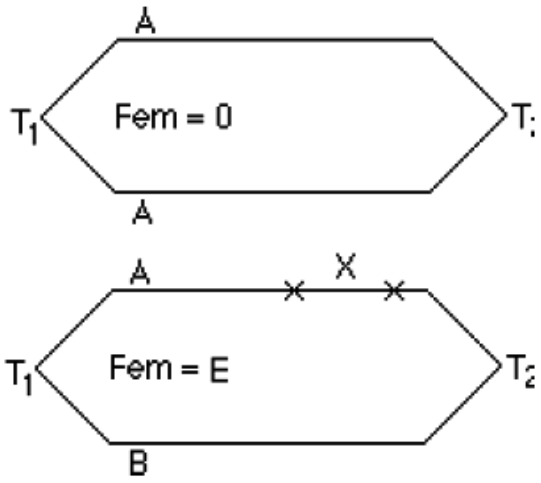


Figura 19: Ilustración de la primera ley de los termopares

II. *Ley de metales intermedios*. Si en un circuito de varios conductores la temperatura es uniforme desde un punto de soldadura A a otro punto B, la suma algebraica de todas las fuerzas electromotrices es totalmente independiente de los conductores metálicos intermedios y es la misma que si se pusieran en contacto directo A y B.

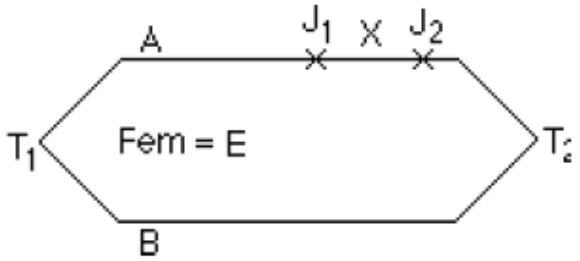


Figura 20: Ilustración de la segunda ley de los termopares

III. *Ley de las temperaturas sucesivas*. La f.e.m. generada por un termopar con sus uniones a las temperaturas T1 T3 es la suma algebraica de la f.e.m. del termopar con sus uniones a T1 T2 de la f.e.m. del mismo termopar con sus uniones a las temperaturas T2 T3.

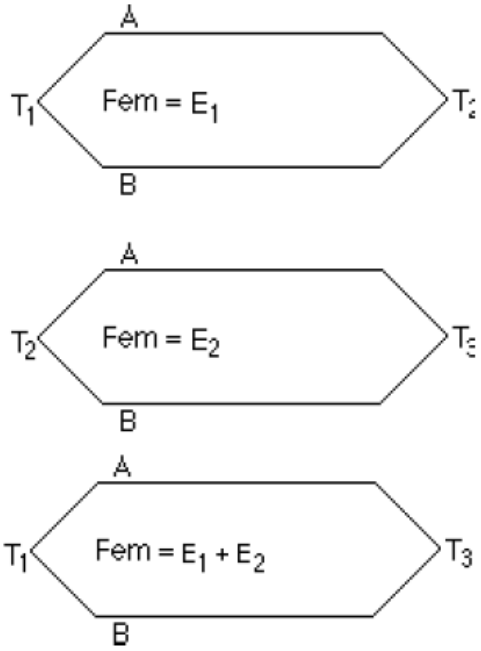


Figura 21: Ilustración de la III ley de los termopares

De acuerdo al rango de operación y a la combinación de aleaciones que los componen los termopares se clasifican dentro de los siguientes tipos [web5]:

Tipo	Materiales
J	Hierro-Constantán (Galga 14).
T	Cobre-Constantán (Galga 20).
K	Cromel-Alumel (Galga 14).
R	Platino/Rodio(13%)-Platino.
S	Platino/Rodio(13%)-Platino.
J*	Hierro-Constantán (Galga 8).

Tabla 3: Tipos de termopares según el material

La elección de un termopar para una aplicación industrial, depende en términos generales del rango de operación al que será sometido, a la exactitud requerida y al costo del mismo.

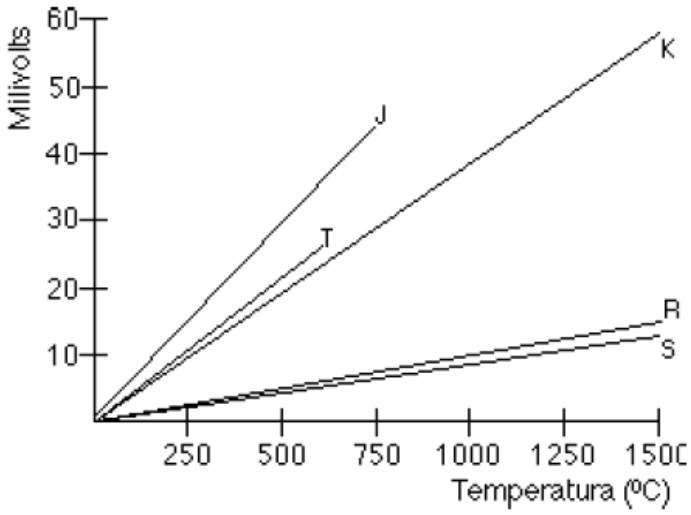


Figura 22: Curva temperatura-voltaje de los diferentes tipos de termopares

Los termopares a pesar de ser sensores lineales no están exentos de desventajas que limitan su uso. En primer lugar es necesario apuntar que tienen una relación señal a ruido (SNR) muy baja, es decir, son muy susceptibles a sufrir alteraciones por ruido. Esto es consecuencia lógica, ya que al tratarse de dos conductores, cuando están instalados cerca de campos magnéticos fuertes tendrán voltajes inducidos cuando su longitud sea larga. Es por ello que requieren de un acondicionador que incluya filtros que eliminen los voltajes inducidos de altas frecuencias con un transmisor adecuado cerca del lugar de medición.

Otra desventaja a la hora de utilizar termopares es que para aplicar el efecto Seebeck a la medida de temperatura, es necesario mantener una de las uniones a una temperatura de referencia. Una solución consiste en disponer la unión de referencia en hielo fundente. [web11]

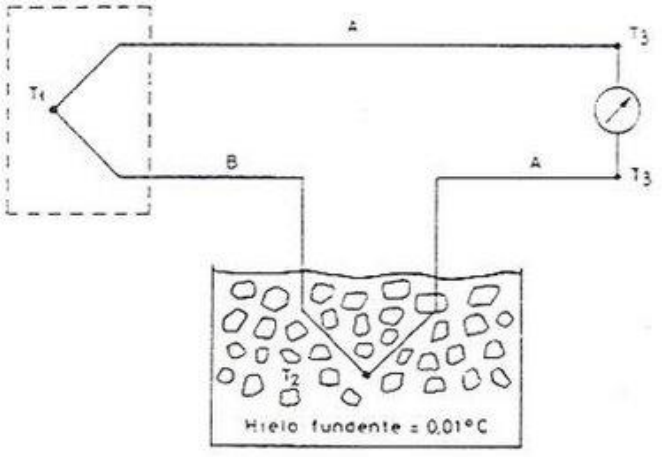


Figura 23: Método del hielo fundente para la medida con termopares

Es una solución de gran exactitud y facilidad de montaje pero es de difícil mantenimiento y coste alto.

La solución indicada en la figura 23 permite emplear un hilo de conexión más económico (como por ejemplo el cobre). Si bien sigue siendo una solución cara por la necesidad de mantener una temperatura de referencia constante. Si el margen de variación de la temperatura ambiente es menor que la resolución deseada puede dejarse la unión de referencia simplemente al aire. En caso contrario se emplea la denominada compensación electrónica de la unión de referencia.

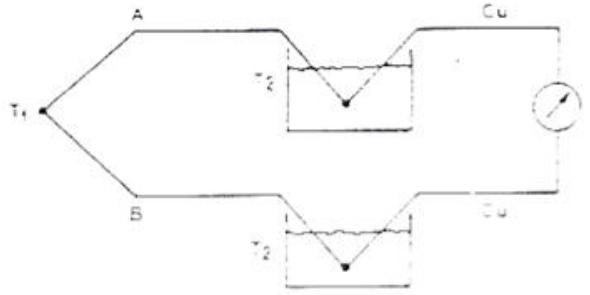


Figura 24: Método de medida de temperatura en termopares

La compensación electrónica consiste en dejar que la unión de referencia sufra las variaciones de la temperatura ambiente, pero esta se detecta con otro transductor de temperatura dispuesto en la vecindad de la unión de referencia, y se resta una tensión igual a la generada en la unión fría. La tensión de alimentación del puente debe de ser estable

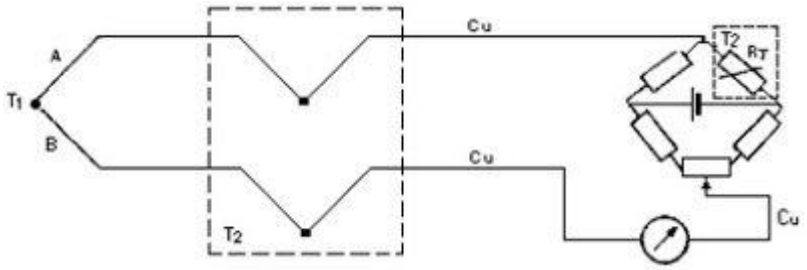


Figura 25: Método de compensación electrónica

La ventaja de este tipo de sensores es su precio y sencillez y resistencia puesto que no es más que dos metales unidos en el punto de medición. Como estos metales pueden ser simples alambres, la inercia térmica es prácticamente nula. Así mismo, son los únicos sistemas capaces de medir temperaturas por encima de los 400ºC gracias a la amplia gama de metales utilizables.

Es común encontrar arreglos de termopares ideados para obtener mejores respuestas en cuanto a la magnitud de fem termo-generada en la unión. Cuando la señal de salida por el termopar es baja y su relación señal a ruido debe ser mejorada entonces se emplea el concepto de termopila que se logra al conectar en serie varios termopares como se aprecia en la figura 26.

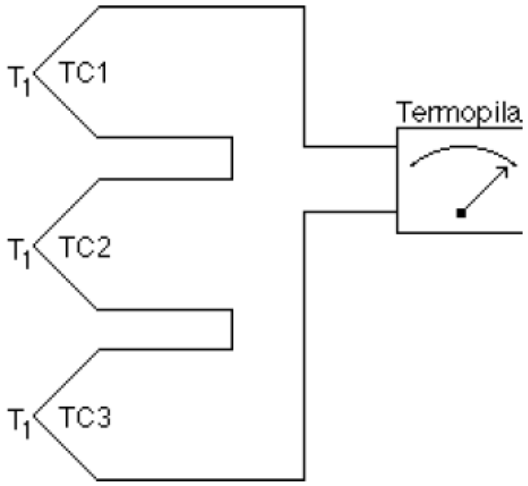


Figura 26: Termopila

Sensores IC de circuitos integrados

Los sensores de circuitos integrados resuelven el problema de la linealidad y ofrecen altos niveles de rendimiento. Son, además, relativamente económicos y bastante precisos a temperatura ambiente.

Sin embargo, los sensores de IC no tienen tantas opciones de configuraciones del producto o de gama de temperaturas, y además son dispositivos activos, por lo que requieren una fuente de alimentación.

Pirómetros de radiación

Un pirómetro es un aparato de de medición de la temperatura de la radiación emitida por un cuerpo por lo que no es necesario el contacto directo del medidor con el cuerpo a medir con sus consecuentes ventajas en cuanto a facilidad y comodidad.

Un pirómetro no es más que un sensor de luz pero que, en vez de detectar la luz visible, detecta la luz infrarroja. Como se puede apreciar en la figura 27 el infrarrojo está justo por encima de la luz visible y este tiene una frecuencia distinta dependiendo de la temperatura a la que esté el cuerpo a medir y de las propiedades del material del mismo.

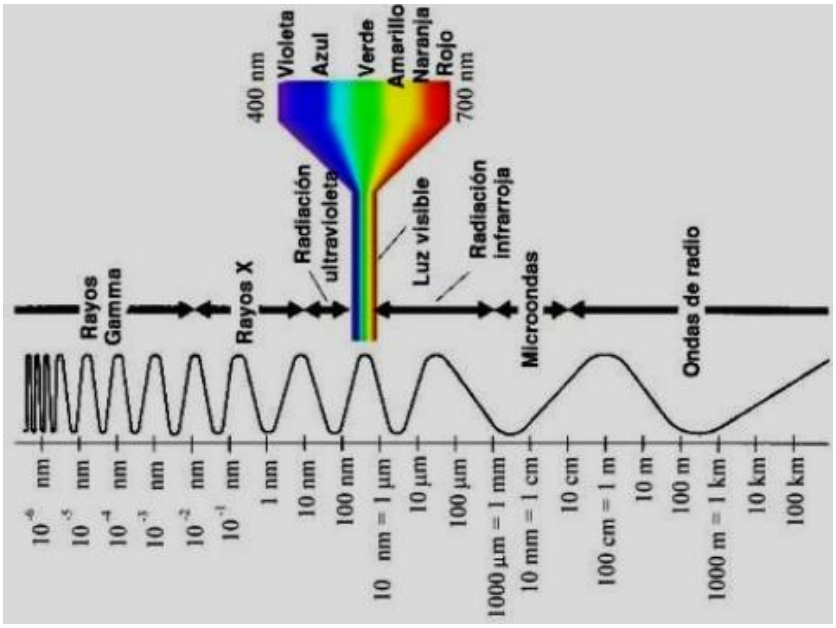


Figura 27: Espectro radiológico

Un pirómetro es capaz de medir la temperatura de cualquier cuerpo.

Este sistema de medición de temperatura tiene en su ventaja su gran inconveniente. Al medir radiaciones estas se comportan de forma característica de forma que se reflejan en unos materiales y en otros no por lo que las mediciones que te de un pirómetro no son siempre fiables. Por ejemplo, si quieres medir la temperatura de un espejo perfecto no podrás usar un pirómetro pues solo conseguirás medir la temperatura de lo que esté reflejándose en el espejo

2.2.3 Transductor utilizado: Melexis 90614

Este es el sensor que se ha utilizado en el prototipo diseñado, y como se ha comentado anteriormente, forma parte del grupo de pirómetros de radiación.

La radiación infrarroja es una parte de la luz solar y puede descomponerse reflejándose a través de un prisma. Esta radiación posee energía. Esto hace posible definir la energía en relación con curvas de emisión de un cuerpo negro. Los objetos con una temperatura por encima del punto cero absoluto irradian energía. La cantidad de energía crece de manera proporcional a la cuarta potencia de la temperatura. Este concepto es el principio básico de la medición de la temperatura por medio de infrarrojos.

El sensor a utilizar es un sensor infra-rojo de temperatura para medidas sin contacto.

Figura 28: MLX90614

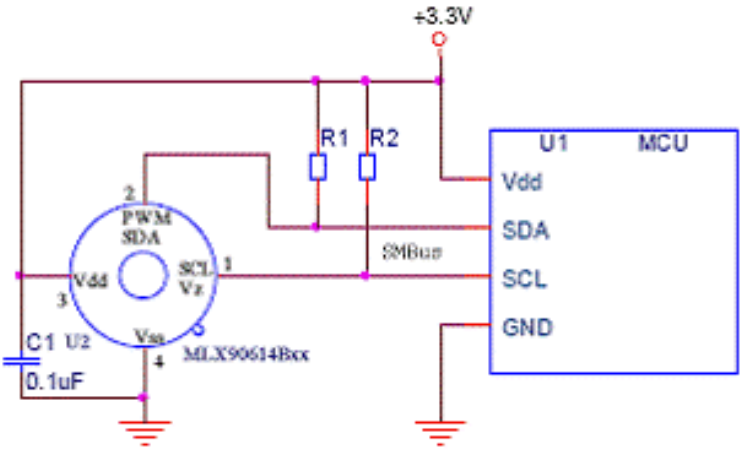


Figura 29: Esquema del conexionado del sensor MLX90614

Características generales.

- Tamaño pequeño y bajo coste.
- Fácil de integrar.
- Calibrado en un ancho de temperatura: -40 hasta 125 °C del sensor de temperatura y -70 hasta 380 °C para la temperatura del objeto.
- Precisión del 0.5°C sobre el ancho del rango de temperatura
- Gran precisión en la calibración.
- Resolución de medida de 0.02°C
- Versiones para una sola zona o para doble zona.
- SMBus interfaz digital compatible para lecturas rápidas de temperatura y para crear redes de sensores.
- Salida PWM adaptable para lecturas continuas.
- Disponible en versiones de 3v y 5v.
- Adaptación simple para aplicaciones de 8 a 16V.
- Modo ahorro de energía.
- Opciones de paquetes diferentes para aplicaciones y versatilidad de medidas.

Ejemplos de aplicación:

- Medidas de temperatura sin contacto de gran precisión.
- Elemento sensible a la temperatura para aires acondicionados en residencias, comercios y construcciones industriales.
- Detector del ángulo de viento.
- Control de temperatura industrial de partes móviles.
- Control de temperatura en impresoras y fotocopiadoras.
- Control de temperatura para aplicaciones en el hogar.
- Cuidados de salud.
- Detector de movimiento
- Monitorización de stock vivo.
- Control de temperatura en zonas múltiples, unos 100 sensores pueden ser leídos por dos vías comunes.
- Medición de la temperatura corporal.

El MLX90614 es un termómetro infra-rojo para medidas de temperatura sin contacto. Ambos, el chip detector sensible a la temperatura y el acondicionador de señal están integrados en la misma cápsula TO-39.

Gracias a su amplificador con ruido bajo, a su 17-bit ADC (Analogic-Digital Conversor, conversor analógico digital) y a la potente unidad DSP (Digital Signal Processor, procesador de señal digital), se consiguen una gran precisión y resolución del termómetro.

El termómetro viene de fábrica calibrado con una modulación PWM (pulse-width modulation, modulación por ancho de pulso) digital y con salida SMBus (System Management Bus, bus de administración del sistema).

Como estándar, la PWM de 10 bits, está configurada para transmitir continuamente las medidas de temperatura en un rango de -20 a 120 ºC, con una salida de resolución de 0,14 ºC y el por defecto es SMBus.

El MLX90614 está construido desde dos chips desarrollados y manufacturado por MELEXIS:

- El termómetro detector infra-rojo MLX81101
- El acondicionador de señal MLX90302, especialmente diseñado para procesar salidas del sensor IR.

El dispositivo está disponible en un paquete estándar industrial TO-39 (encapsulado).

El objeto calculado y la temperatura ambiente están disponibles en la RAM del MLX90302 con una resolución del 0,01 ºC. Son accesibles por dos cables serie compatibles con el protocolo SMBus (0.02 ºC de resolución) o vía la salida de 10-bit PWM (Modulación por ancho de pulso) del dispositivo.

2.3 Resistencias Pull-up

Las resistencias Pull-up se usan en circuitos lógicos electrónicos para asegurar que las entradas a los sistemas lógicos se establecen la los niveles lógicos esperados si los dispositivos externos están desconectados o en estados de alta impedancia. También se usan en la interfaz entre dos dispositivos lógicos de diferentes tipos que posiblemente operen a distintos niveles de tensión de alimentación. [web13]

La idea de estos resistores está relacionada con los tres estados lógicos. Las resistencias pull-up hacen que el voltaje del cable no conectado a la tensión de alimentación sea 5v (pull-up) cuando el resto de componentes de la línea estén inactivos.

Esto es así porque cuando todas las otras conexiones de la línea están inactivas, se dice que están en alta impedancia y actúan como si estuviesen desconectadas. Como los otros componentes del circuito actúan como si estuviesen desconectados, el circuito en sí actúa como si estuviese desconectado y no circula corriente por la línea.

Sin embargo cuando otro componente de la línea se activa, la línea entera se pondrá a ese nivel, ya que es común a todos los componentes (todos los componentes de la línea deben estar al mismo nivel de voltaje).

Esto es importante porque a pesar de que los componentes estén a alta impedancia, actuando como si estuviesen desconectados, pueden tomar el voltaje de la línea como una entrada y esto puede afectar a su valor.

Es decir, las resistencias pull-up no son más que resistencias que se conectan entre una señal lógica y la fuente de alimentación, para que aquella no quede en estado flotante.

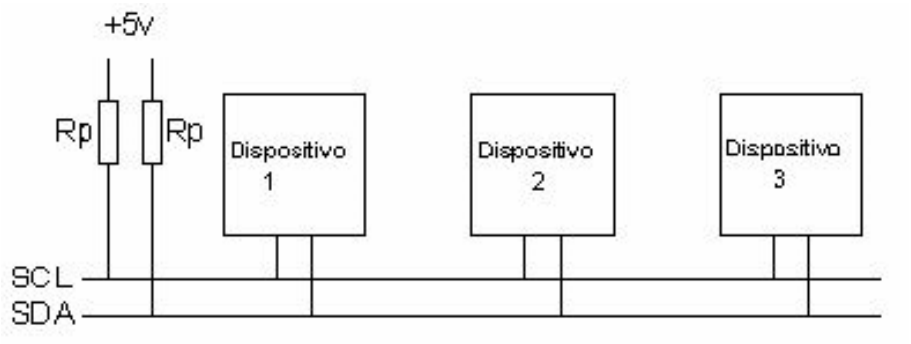


Figura 30: Ejemplo de aplicación de resistencias pull-up (Rp)

2.3.1 Resistencia Pull-up dinámica LTC1694 utilizada

La LTC1694 es un resistor pull-up dinámico y dual para realzar la velocidad y la fiabilidad de la transmisión de datos bajo todas las especificaciones y condiciones del bus de datos SMBus. Es también compatible con la tecnología I2C de Philips. [datasheet LTC1694]

Estos resistores permiten la conexión de varios dispositivos o una interconexión más larga y capacitada, sin comprometer las funciones del bus.

Las características generales del dispositivo son:

- Mejora la velocidad de comunicación del bus
- Asegura la integridad de los datos con varios dispositivos conectados al bus
- Autodetección del estado de baja potencia o standby.
- Pequeño tamaño
- Mejora el margen de ruido

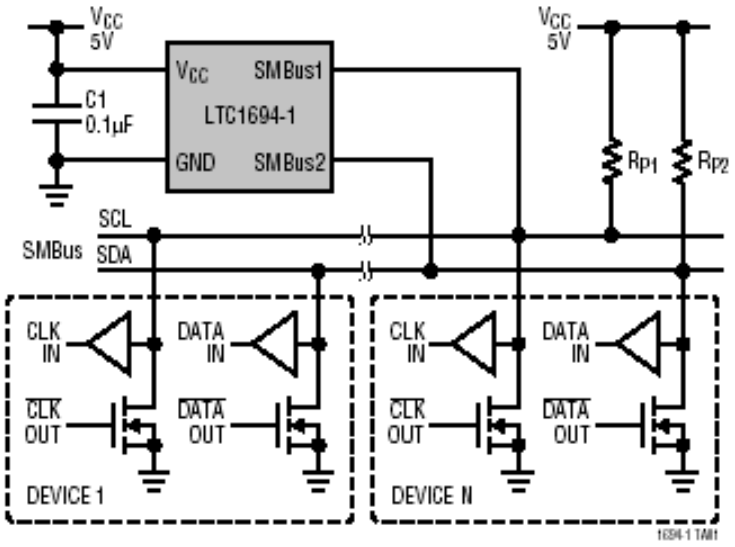


Figura 31: ejemplo de conexionado y aplicación de la LTC1694

2.4 Microprocesadores

Desde el punto de vista funcional, un microprocesador es un circuito integrado que incorpora en su interior una unidad central de proceso (CPU) y todo un conjunto de elementos lógicos que permiten enlazar otros dispositivos como memorias y puertos de entrada y salida (I/O), formando un sistema

completo para cumplir con una aplicación específica. Para que el sistema pueda realizar su labor debe ejecutar paso a paso un programa que consiste en una secuencia de números binarios o instrucciones, almacenándolas en uno o más elementos de memoria, generalmente externos al mismo. [web14]

El microprocesador o simplemente procesador, es el circuito integrado más importante, de tal modo, que se le considera el cerebro de una computadora. Está constituido por millones de transistores integrados. Puede definirse como chip, un tipo de componente electrónico en cuyo interior existen miles o en ocasiones millones, según su complejidad, de elementos llamados transistores cuyas interacciones permiten realizar las labores o funciones que tenga encomendado el chip.

Así mismo, es la parte de la computadora diseñada para llevar a cabo o ejecutar los programas. Éste ejecuta instrucciones que se le dan a la computadora a muy bajo nivel realizando operaciones lógicas simples, como sumar, restar, multiplicar o dividir. Se ubica generalmente en un zócalo específico en la placa o tarjeta madre y dispone para su correcto y estable funcionamiento de un sistema de refrigeración (generalmente de un ventilador montado sobre un disipador de metal térmicamente muy conductor).

Su "velocidad" se determina por la cantidad de operaciones por ciclo que puede realizar y los ciclos por segundo que desarrolla: también denominada frecuencia de reloj. La frecuencia de reloj se mide Hertzios, pero dado su elevado número se utilizan los múltiplos megahertzio o gigahertzio.

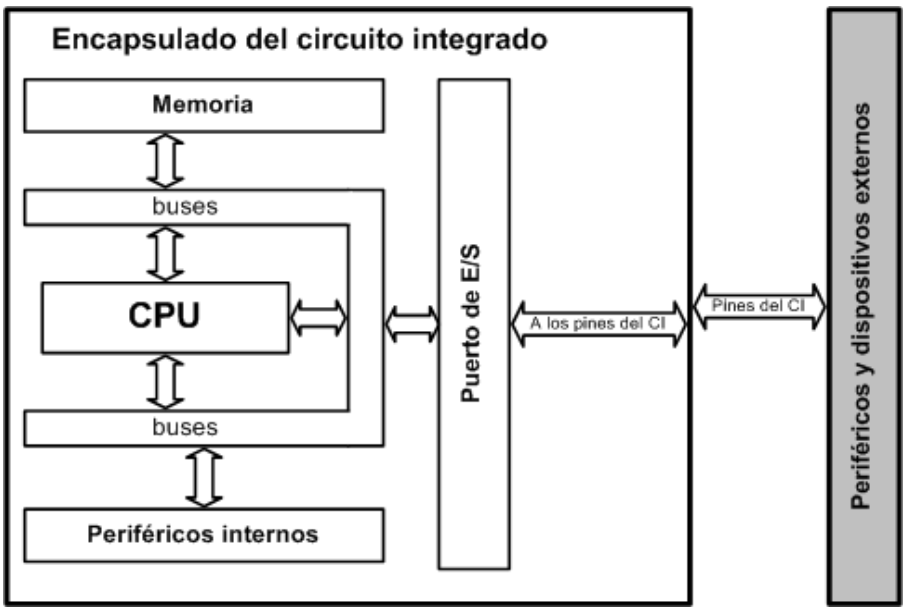


Figura 32: Esquema de un microprocesador

2.4.1 Microprocesador utilizado: ARM7

Se denomina ARM (Advanced RISC Machines) a una familia de microprocesadores diseñados por la empresa Acorn Computers y desarrollados por Advanced RISC Machines Ltd., una empresa derivada de la anterior.

El ARM7 es un microprocesador moderno de bajo coste orientado a sistemas empotrados, siendo el LPC2129 el micro controlador en el cual se basa la CPU ARM7TDMI-S.

El micro controlador también incorpora:

- 256 kb de memoria flash.
- 16kb de RAM estática.
- Mecanismo de emulación por JTAG, ISP ICE-RT.
- Controlador de interrupciones vectorizadas de prioridad configurable
- 2 puertos serie asíncronos (UART0, UART1).
- Puertos serie síncronos (Fast I2C y SPI)
- 2 temporizadores de 32 bits
- 4 canales ADC de 10 bits.
- 46 pines de propósito general tolerantes a 5v.
- 9 entradas de IRQ externa con disparo por nivel o flanco.
- 2 modos de bajo consumo.
- 2 canales de CAN-bus.

Todo periférico va a ser visto por la CPU como un conjunto de registros que podrán ser de tres tipos:

- De datos: los que van a contener los datos que se van a utilizar en el periférico y que se comunicarán a/desde la CPU.
- De estado: los que van a contener información sobre el estado en que se encuentra el periférico.
- De control: los que se van a escribir para configurar el periférico.

Para acceder a los registros, la CPU utilizará dos formas:

- Mediante instrucciones especiales de Entrada/Salida.
- Como si accediesen a una dirección de memoria (mapeado de memoria de los periféricos).

En el caso de la CPU contenida en el LPC2129 y de todos los sistemas basados en la arquitectura ARM, para acceder a los periféricos se utiliza el mapeado en memoria, por lo tanto, los registros de los periféricos de E/S van a verse como direcciones de memoria.

Algunas características de este tipo de comunicación con los periféricos son:

- La mayor parte de los periféricos están conectados al bus VPB (very large scale integration peripheral bus, es decir, bus periférico de muy larga escala de integración) por tanto, mapeados en las direcciones 0xE000 0000 – 0xE0F FFFF.

- La mayoría son registros de 32 bits.

- La conexión de cada periférico se hace a través de los mismos pines que los de propósito general:
    - Los pines del GPIO serán "multifunción".
    - Su función se seleccionará con los registros:
        PINSEL0 (0xE002 C000): Puerto 0 – pines 0…15
        PINSEL1 (0xE002 C004): Puerto 0 – pines 16…31
        PINSEL2 (0xE002 C014): Puerto 1 – pines 16…31

El mapa de memoria del LPC2129 está compuesto por:

- 256KB Flash
    - 0x0000 0000 – 0x0003 FFFF
- 16KB RAM estática
    - 0x4000 0000 – 0x4000 3FFF
- Bloque de arranque
- Periféricos conectados al bus VPB
    - 0xE000 0000 – 0xEFFF FFFF
    - Realmente solo esta mapeado en la región 0xE000 0000 – 0xE01F FFFF
- Periféricos conectados al bus AHB
    - 0xF000 0000 – 0xFFFF FFFF
    - Realmente solo esta mapeado en la región 0xFFE0 0000 – 0xFFFF FFFF

El subsistema de reloj que integra el microprocesador tiene las siguientes características:

- El Oscilador externo soporta cristales (Xtal) de frecuencia de oscilación entre 1MHz y 30 MHz (Fosc)

- La frecuencia interna del micro (cclk) se obtiene de Fosc:
    - Si el PLL interno no está funcionando:
      cclk = Fosc
    - Si se está utilizando el PLL interno:
      Fosc debe estar entre 10 y 25 MHz
      cclk = M*Fosc (siento M un numero entero entre 1 y 32 y el cclk
      máximo de 60MHz).

* *La configuración del PLL se realiza con los registros llamados PLLCON, PLLCFG, PLLSTAT, PLLFEED*

- La frecuencia del bus VPB (pclk) puede ser:
    - Pclk = cclk/4 (valor por defecto)
    - Pclk = cclk /2
    - Pclk =cclk

Algunos ejemplos de equipos comerciales en los cuales se incorpora el microprocesador ARM7 son:

- RIM Blackberry 7230
- Motorola MPx
- Nokia 6630 Mobile Phone
- Holux GR-230 Bluetooth GPS Receiver

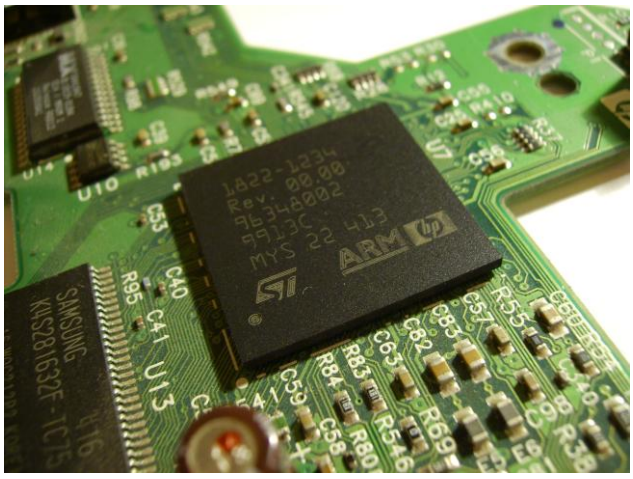


Figura 33: Imagen de un microprocesador ARM en una placa

2.5 Tecnología de comunicación I2C

I2C son las siglas para "Inter – Integrated Circuit". I2C es un protocolo síncrono que permite a un dispositivo maestro iniciar comunicación con un dispositivo esclavo. [web15]

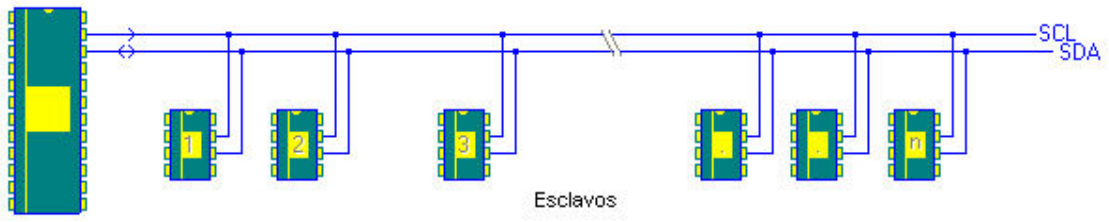


Figura 34: Esquema de un sistema de comunicación I2C con varios esclavos

La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (pulsos de reloj) y la otra se utiliza para intercambiar datos.

En la comunicación serie se transmite la información bit a bit, se divide la información en palabras y cada palabra se transmite bit a bit. Tanto para indicar la existencia de cada bit como para indicar en inicio de una palabra se precisan métodos y puntos de sincronismo.

Atendiendo al sincronismo de bit se darán:

- La comunicación síncrona, cuando una señal de reloj indique la validez del bit transmitido.
- La comunicación asíncrona cuando no existe dicha señal de reloj, por lo que la temporización de emisor y receptor se realiza por medios independientes.

Descripción de las señales del bus I2C

- **SCL** (System Clock) es la línea de los pulsos de reloj que sincronizan el sistema.
- **SDA** (System Data) es la línea por la que se mueven los datos entre los dispositivos.
- **GND** (Masa) común de la interconexión entre todos los dispositivos "enganchados" al bus.

Las líneas SDA y SCL están conectadas a todos los dispositivos en el bus I2C. Las líneas SDA y SCL son del tipo drenador abierto. Lo que quiere decir es

que el chip puede poner su salida a nivel bajo, pero no a alto. Para que la línea sea capaz de ir a un estado alto, se debe utilizar resistencias "pull-up" (que ya se han explicado con anterioridad) a la fuente de 5 voltios (Vdd) (ver Figura 35). Debe haber una resistencia desde la línea SCL a los 5 voltios y otra de la línea SDA a la línea de 5voltios. Sólo se necesita un juego de resistencias pull-up para todo el bus I2C, no para cada dispositivo como se ilustra abajo. En la Figura 35 se muestra de que manera van conectadas las resistencias en las líneas de reloj y datos.

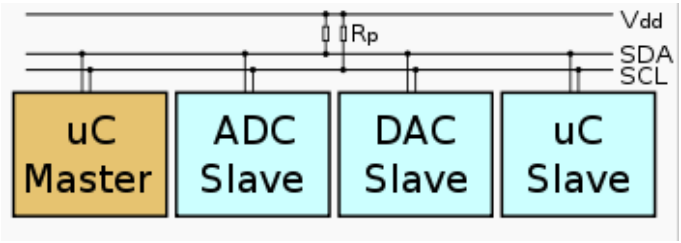


Figura 35: Resistencias de pull-up en el bus I2C

Otras características:

- Utiliza niveles TTL (0, +5voltios).
- Permite el modo de operación multimaestro.
- Dirección única de comunicación.
- La validación del bit de datos es con SCL a nivel alto.
- La trama se delimita con la condición START y STOP.
- Permite hasta 128 periféricos conectados a la misma línea.

2.5.1 Maestros y esclavos

Los dispositivos en el bus I2C son maestros o esclavos. Los dispositivos conectados al bus I²C tienen una dirección única para cada uno. El dispositivo maestro inicia la transferencia de datos y además genera la señal de reloj, pero no es necesario que el maestro sea siempre el mismo dispositivo, esta característica se la pueden ir pasando los dispositivos que tengan esa capacidad. Esta característica hace que al bus I²C se le denomine bus multimaestro.

El maestro es siempre el dispositivo que conduce la línea de reloj de SCL. Los esclavos son los dispositivos que responden al maestro. Un esclavo no puede iniciar una transferencia sobre el bus I2C, solo un maestro puede hacerlo. Puede haber, y por lo general así es, múltiples esclavos en el bus I2C; sin embargo, es normal que solo exista un maestro.

Tanto el esclavo como maestro pueden transferir datos sobre el bus I2C, pero esa transferencia es siempre controlada por el maestro.

2.5.2 Protocolo de comunicación

Habiendo varios dispositivos conectados sobre el bus, es lógico que para establecer una comunicación a través de él se deba respetar un protocolo. Lo más importante, como se ha dicho antes: existen dispositivos **maestros** y dispositivos **esclavos y s**ólo los dispositivos maestros pueden iniciar una comunicación. [web16]

La condición inicial, de **bus libre**, es cuando ambas señales están en estado lógico alto. En este estado cualquier dispositivo maestro puede ocuparlo, estableciendo la condición de **inicio** (start). Esta condición se presenta cuando un dispositivo maestro pone en estado bajo la línea de datos (SDA), pero dejando en alto la línea de reloj (SCL).

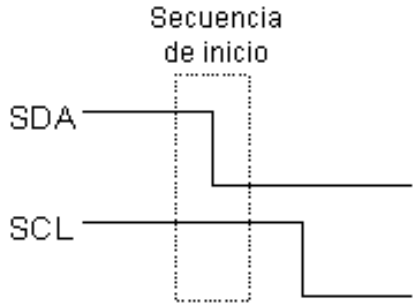


Figura: condición de inicio comunicación I2C

El primer byte que se transmite luego de la condición de inicio contiene siete bits que componen la dirección del dispositivo que se desea seleccionar, y un octavo bit que corresponde a la operación que se quiere realizar con él (lectura o escritura).

Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits (A0-A6) está presente en el bus, éste contesta con un bit en bajo, ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro. Este bit de **reconocimiento** (ACK) en bajo le indica al dispositivo maestro que el esclavo **reconoce** la solicitud y está en condiciones de comunicarse. Aquí la comunicación se establece en firme y comienza el intercambio de información entre los dispositivos.

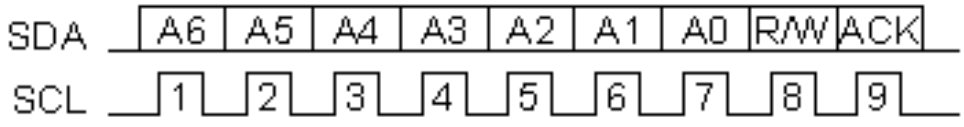


Figura 36: Bits transmitidos en una comunicación I2C

Si el bit de **lectura/escritura** (R/W) fue puesto en esta comunicación a nivel lógico bajo (escritura), el dispositivo maestro envía datos al dispositivo esclavo. Esto se mantiene mientras continúe recibiendo señales de reconocimiento, y el contacto concluye cuando se hayan transmitido todos los datos.

En el caso contrario, cuando el bit de lectura/escritura estaba a nivel lógico alto (lectura), el dispositivo maestro genera pulsos de reloj para que el dispositivo esclavo pueda enviar los datos. Luego de cada byte recibido el dispositivo maestro (quien está recibiendo los datos) genera un pulso de **reconocimiento**.

El dispositivo maestro puede dejar libre el bus generando una condición de **parada** (o detención; stop en inglés).

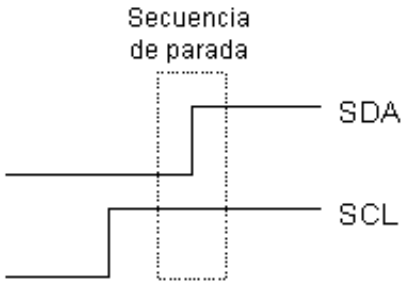


Figura 37: Condición de parada del bus I2C

Si se desea seguir transmitiendo, el dispositivo maestro puede generar otra condición de inicio en lugar de una condición de parada. Esta nueva condición de inicio se denomina "inicio reiterado" y se puede emplear para direccionar un dispositivo esclavo diferente o para alterar el estado del bit de lectura/escritura.

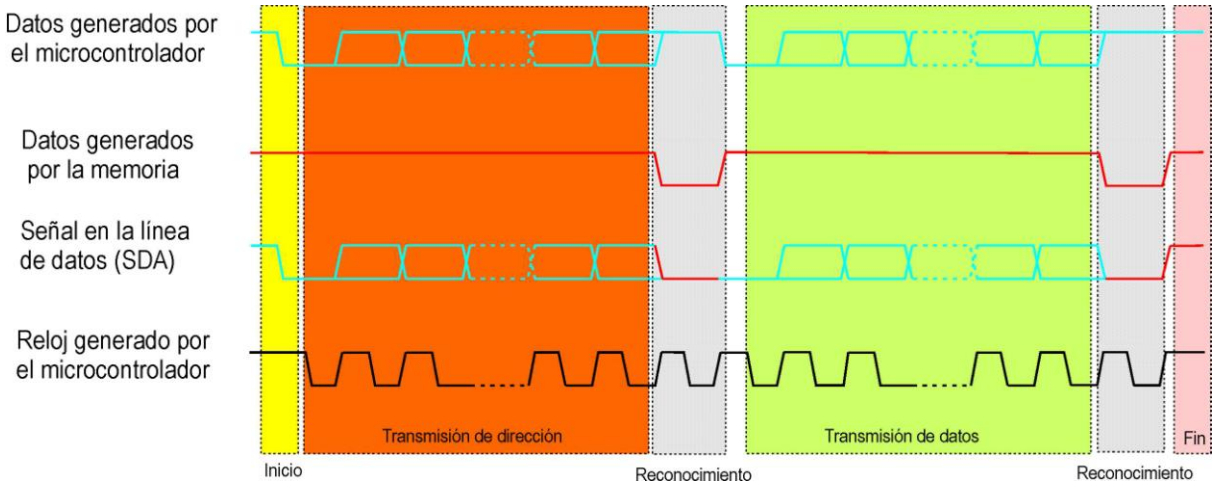


Figura 38: Comunicación completa en el bus I2C

2.5.3 Registros de configuración I2C para el ARM7

La interfaz periférica I2C está compuesta por 7 registros. El registro de control son realmente dos I2CONSET y I2CONCLR, utilizados para poner a uno y a cero respectivamente los bits correspondientes:

- Bit 2 (AA), Assert Acknowledge: Si está a 1, cuando se reciba un carácter, se dará un ACK. Si es un 0, no da ACK.
- Bit 3 (SI), Flag de Evento/Interrupción. Se activa con la utilización del I2C:
    - Se limpia con un 1 en el bit del I2CONCLR.
    - Al limpiarlo se da la señal de que se envíe/reciba el dato.
- Bit 4 (STO): Escribir un 1 provoca la emisión de una condición de STOP. Se limpia automáticamente.
- Bit 5 (STA): Escribir un 1 provoca volver a modo Maestro y mandar una condición de START.

Otros registros importantes son I2STAT,el registro de estados, que devolverá diferentes códigos dependiendo de lo que esté ocurriendo en el bus; el I2DAT que es el registro de datos, el cual almacenará los datos a ser transmitidos o recibidos, es bidireccional; y el I2ADR, el registro de dirección de esclavo.

Los otros dos registros son I2SCLH y I2SCLL que se utilizan para determinar la tasa de bits, es decir, el número de bits que se transmiten por unidad de tiempo.

Name	Description	Access	Reset Value*	Address
I2CONSET	I^2C Control Set Register	Read/Set	0	0xE001C000
I2STAT	I^2C Status Register	Read Only	0xF8	0xE001C004
I2DAT	I^2C Data Register	Read/Write	0	0xE001C008
I2ADR	I^2C Slave Address Register	Read/Write	0	0xE001C00C
I2SCLH	SCL Duty Cycle Register High Half Word	Read/Write	0x04	0xE001C010
I2SCLL	SCL Duty Cycle Register Low Half Word	Read/Write	0x04	0xE001C014
I2CONCLR	I^2C Control Clear Register	Clear Only	NA	0xE001C018

Tabla 4: Registros del I2C en un microprocesador LPC2119

Capítulo 3

Descripción del prototipo

Puesto que los diferentes elementos del sistema ya se han descrito con anterioridad, en los siguientes apartados nos limitaremos a detallar el procedimiento que se ha seguido para el conexionado de los diferentes dispositivos así como el protocolo de comunicación que siguen y los valores y métodos de configuración de los mismos.

3.1 Características

El objetivo de éste proyecto es desarrollar un sistema de control de posición para una cámara de iris. El sistema es un montaje alta-azimutal, es decir, en dos ejes, que se controla a través de la detección de la temperatura del sujeto.

El dispositivo se ha diseñado utilizando una placa del microprocesador ARM7, que ya se ha descrito con anterioridad en este mismo documento.

Para la comunicación entre los distintos dispositivos se ha estudiado el bus I2C de Philips integrado en el microprocesador LPC2129 que hemos utilizado.

3.1.1 Elementos del sistema

El prototipo consta de los siguientes elementos:

- Sensor de temperatura MLX90614
- Controlador TMC222 para un motor PaP
- Resistencias Pull-up LTC1694
- Placa de evaluación MCB2100

De los elementos enumerados, el único que queda por describir es la placa de evaluación MCB2100 que detallaré en el apartado de descripción del hardware.

3.1.2 Herramientas utilizadas

Para llevar a cabo el diseño del prototipo se han utilizado diferentes herramientas dependiendo de si se trataba de la parte de software o del hardware del sitema.

Desarrollo del software

El software de comunicación y de configuración de los distintos dispositivos se ha descrito con el programa uVision3 del fabricante Keil.

Con esta herramienta podemos crear programas, simularlos y compilarlos en un entorno de desarrollo con interfaz gráfica (IDE). Este entorno nos permite utilizar lenguaje C y está específicamente desarrollado para microprocesadores como el ARM7 además de otros muchos.

Este entorno nos permitirá simular la comunicación a través del I2C.

Figura 39: Logotipo del software uVision3

Desarrollo del hardware

El diseño de la placa se ha realizado utilizando la herramienta Orcad de diseño asistido por computador, la cual es ampliamente utilizada en el diseño de circuitos electrónicos.

De entre las aplicaciones que forman parte de la herramienta hemos utilizado:

- **Orcad Capture** para la realización del esquemático del circuito electrónico, es decir, para diseñar las conexiones entre los distintos elementos. También se ha utilizado para la creación de nuevas librerías para los componentes sensor, controlador y resistencias pull-up.

- **Orcad Layout** para la realización de la placa del circuito impreso. Además, dentro de esta aplicación, se ha utilizado el editor de huellas para la creación de las huellas del sensor, el controlador y la resistencia de pull-up.

Figura 40: Logotipo del software Orcad de Cadence.

3.2 Descripción del software

El software que se ha desarrollado ha sido el de configuración del sistema bajo el protocolo de comunicación I2C, tanto del sensor como del controlador del motor.

3.2.1 Software del sensor

El programa desarrollado para el sensor de temperatura MLX90614 realiza la configuración de diferentes registros del mismo como son los concernientes a la temperatura del ambiente, la temperatura del objeto, la dirección del esclavo etc. También se han desarrollado funciones para la lectura de registros y la activación del modo *sleep* del sensor, que es un modo de baja energía.

En los apartados siguientes se procederá a detallar los códigos, la comunicación y la configuración del sensor.

Descripción de las memorias del sensor

El sensor MLX90614 sólo puede ser utilizado como esclavo en la comunicación, así, generalmente el maestro inicia el envío de datos seleccionando un esclavo mediante su dirección (Slave Address).

El MLX90614 contiene una memoria RAM de 32x17 bits que tan sólo puede ser escrita y de la cual tan sólo unos pocos registros contienen información útil para el usuario. Estos registros son los detallados en la Tabla 5:

Name	Address
Datos de ambiente del sensor	0x03
IR datos sensor 1	0x04
IR datos sensor 2	0x05
Temperatura ambiente linealizada [Ta]	0x06
Temperatura objeto 1 linealizada [Tobj1]	0x07
Temperatura objeto 2 linealizada [Tobj2]	0x08

Tabla 5 Registros RAM del sensor MLX90614

El sensor también cuenta con una memoria EEPROM de 32x16 bits la cual puede ser leída y escrita. Esta memoria suele contener información sobre la calibración del sensor, configuración del chip etc.

Es esta la que modificaremos para configurar el sensor según las necesidades de las medidas, dentro de unas restricciones. Aunque no todas las celdas pueden ser escritas, sí pueden ser leídas todas. Podremos, por ejemplo, cambiar la dirección de esclavo del sensor.

Los registros de la EEPROM útiles para la configuración del sensor son los mostrados en la Tabla 6:

EEPROM (32X16)		
Name	Address	Write acces
To_{max}	000h	Yes
To_{min}	001h	Yes
PWMCTRL	002h	Yes
Ta range	003h	Yes
Emissivity correction coefficient	004h	Yes
Config Register1	005h	Yes
Melexis reserved	006h	No
...	...	...
Melexis reserved	00Dh	No
SMBus address	00Eh	Yes
Melexis reserved	00Fh	Yes
Melexis reserved	010h	No
...	...	...
Melexis reserved	018h	No
Melexis reserved	019h	Yes
Melexis reserved	01Ah	No
Melexis reserved	01Bh	No
ID number	01Ch	No
ID number	01Dh	No
ID number	01Eh	No
ID number	01Fh	No

Tabla 6: Registros EEPROM del sensor

Descripción de la comunicación

Los comandos necesarios para que se de la comunicación, es decir, para que el sensor responda al microprocesador con la información adecuada son los mostrados en la Tabla 7:

Opcode	Command
000x xxxx*	RAM Access
001x xxxx*	EEPROM Access
1111_0000**	Read Flags
1111_1111	Enter SLEEP mode

Tabla 7: Comandos de comunicación con el sensor

Según se quiera acceder a la memoria RAM, EEPROM, entrar en el modo SLEEP o leer flags, se mandarán al sensor uno u otro comando siguiendo siempre el protocolo de comunicación que se describirá más adelante.

Teniendo en cuenta que:

- Xxxxx representan los 5 bits menos significantes (LSBits) del mapa de memoria que queremos leer/escribir
- Leeremos o escribiremos dependiendo del estado del bit destinado para tal selección y que vernos más adelante cual es.
- El comando Read Flags se comporta como un comando de lectura, la diferencia con el comando de lectura es que el read flags no tiene un bit start repetido. Los flags a leer son:

    - Data[7]- EEBUSY – el anterior acceso a la EEPROM está todavía en proceso. Activo a nivel alto.
    - Data[6]- No se usa.
    - Data[5]- EE-DEAD-EEPROM ha ocurrido un doble error. Activo a nivel alto.
    - Data[4]- INIT – POR la rutina de inicialización está aún activa. Activo a nivel bajo
    - Data[3]- no implementado.
    - Data[2..0] and Data [8..15]- todo ceros.

El lector de flags es una herramienta de diagnóstico pero el MLX90614 puede ser utilizado sin tener en cuenta esta herramienta y nosotros no la tendremos en cuenta ni la utilizaremos.

Una vez descrita la estructura de la memoria interna del sensor así como dónde se almacenan los diferentes datos a configurar y los comandos a utilizar procederemos a explicar cómo se desarrolla la comunicación entre el sensor y el microprocesador, es decir, el protocolo I2C para este caso específico de comunicación.

El protocolo de comunicación general del bus serie I2C se detalla en la Figura 41.

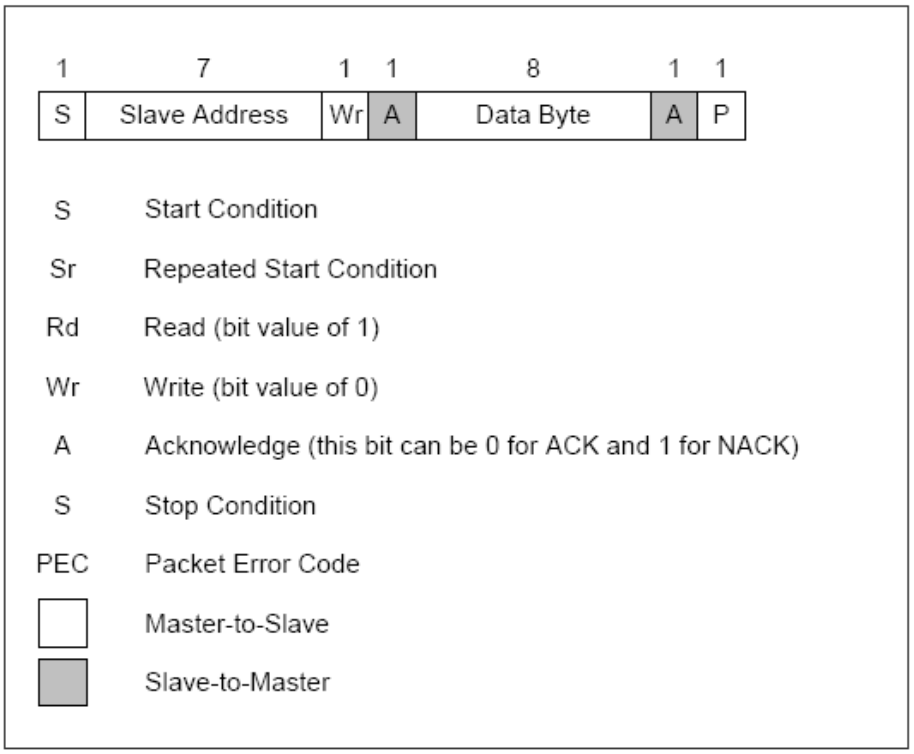


Figura 41: Protocolo de comunicación I2C

Lo primero que el microprocesador ha de enviar al sensor, es decir, al esclavo, es una condición de inicio (*S*) y la dirección del esclavo a la que se dirige (*Slave Adress*), puesto que puede darse el caso se que haya más de un dispositivo en la línea de comunicación. En nuestro caso tendremos el sensor y el motor.

Seguido a la dirección del esclavo aparece un bit (*Wr*) que indica si el esclavo ha de ser leído o ha de escribirse.

Después de cada 8 bits recibidos por el esclavo, tiene lugar un ACK/NACK *(ACK=1 ó ACK=0, NACK)*. Cuando el maestro inicia la comunicación, primero envía la dirección del esclavo, y sólo aquel que reconoce la dirección va a responder con un ACK, el resto permanecerá sin hacer nada. En caso de que el esclavo responda con un NACK en la transferencia de alguno de lo bytes el maestro parará la comunicación y repetirá el mensaje.

Un NACK puede ser recibido también después del PEC (paquete del código del error), esto significa que ha habido un error en la recepción del mensaje y el maestro debe intentar enviar el mensaje de nuevo. El cálculo del PEC incluye todos los bits excepto los de START, REPEATED START, STOP, ACK y NACK. El PEC es un CRC-8 con el polinomio X8+X2+X1+1. En nuestro dispositivo no utilizaremos el PEC.

El bit más significativo de cada byte es el que se transfiere primero.

Comunicación con la RAM:

Como hemos dicho antes, el acceso a la memoria RAM es de sólo lectura. Cuando es leída, los datos se dividen en dos, por ejemplo, para Tobj, la dirección 0x07h barrerá desde 0x27ADh hasta 0x7FFF, ya que la temperatura del objeto va desde -70.01ºC hasta +382.19ºC. El bit más significativo (MSb) leído por la RAM es un flag de error (activo a nivel alto), para la temperatura linealizada (Tobj1, Tobj2, Ta).

El formato de lectura del Bus serie de la memoria RAM se detalla en la Figura 42.

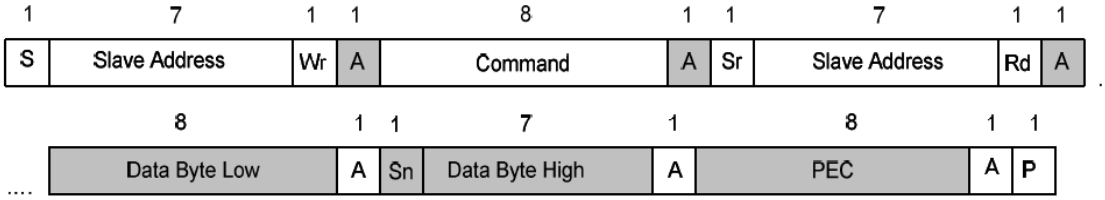


Figura 42: Protocolo de lectura

**P= condición de STOP*
**A= bit de reconocimiento (ACK=1 ó ACK=0)*
**Rd/Wr= son el mismo bit, indica la dirección de la comunicación (1=Read 0=write)*
**S=condición de STOP*
**casillas en blanco=maestro-esclavo*
**casillas en gris=esclavo-maestro*

Los datos que nos interesará leer de la memoria RAM son los determinados en las celdas 0x06, 0x07 y 0x08 de la Tabla 5 de la RAM, que corresponden a las temperaturas del ambiente, del objeto 1 y del objeto 2 respectivamente.

El procedimiento esquematizado que debemos seguir para leer es el siguiente:

* *el ejemplo está hecho para una lectura de la dirección 0x07, es decir, la temperatura del objeto 1, el comando será diferente según el registro a leer/escribir.*

1. Enviar el bit de START.
2. Enviar la dirección del esclavo (0x00*) + 1(bit de lectura).
3. Enviar comando (0b0000_0111).
4. Enviar un repetitivo START_bit.
5. Enviar la dirección del esclavo (0x00*) + 1(bit de lectura).
6. Leer el Dato Byte Low (el maestro debe devolver un bit ACK).
7. Leer el Dato Byte High (el maestro debe devolver un bit ACK).
8. Leer el PEC (el maestro puede devolver un ACK o NACK)

9. Enviar el bit STOP.

Se da a la dirección del esclavo el valor 0x00, puesto que todo dispositivo MLX90614 responderá a ella, pero en caso de que se vaya a utilizar más de un sensor de este tipo en la misma línea de comunicación (bus I2C), deberemos dar una dirección diferente de esclavo a cada dispositivo a través de la escritura en la memoria EEPROM que se detalla más adelante.

Por otro lado, en caso de que queramos leer otras direcciones de la memoria RAM, siempre y cuando tengamos acceso a ellas, tan sólo debemos cambiar, además de la dirección del esclavo al cual nos dirigimos si fuese necesario, el comando a enviar. De este modo, para leer las otras dos direcciones utilizaremos:

- Para 0x06: **0b0000_0110**
- Para 0x08: **0b0000_1000**
- Para 0x07**: 0b0000_0111**

Comunicación con EEPROM:

En la memoria EEPROM podremos hacer operaciones tanto de lectura como de escritura.

- *Lectura de la EEPROM:*

Todas las celdas de la memoria EEPROM pueden ser leídas, pero tan sólo unas pocas contienen información realmente útil.

Las celdas que vamos a utilizar para la configuración de nuestro sensor son las siguientes:

- 000h - corresponde a la Tomax (temperatura del objeto máxima)
- 001h - corresponde a la Tomin(temperatura del objeto mínima)
- 002h - corresponde al PWMCTRL (control de la salida digital)
- 003h - corresponde a Ta range (rango de la temperatura ambiente)
- 004h - corresponde al coeficiente de corrección de la emisividad del objeto.
- 005h - corresponde al Config Register1 (registro de configuración)
- 00Eh - corresponde a la dirección del SMBus (dirección del esclavo)

Los valores con los que vamos a configurar el sensor son:

- Tomax= 382’2ºC
- Tomin=-70ºC
- Ta Range= -40ºC a 125ºC

- Emisividad=1
- Resolución 0'02ºC

Si el acceso a la EEPORM del MLX90614 es en modo lectura el sensor va a responder con 16 bits de datos y 8 bits de error (PEC) sólo si su dirección de esclavo, programada en la EEPROM corresponde con la dirección del esclavo enviada por el maestro, como ya se ha comentado anteriormente, no prestaremos atención a los bits de error.

El formato de lectura del SMBus de la EEPROM es el siguiente:

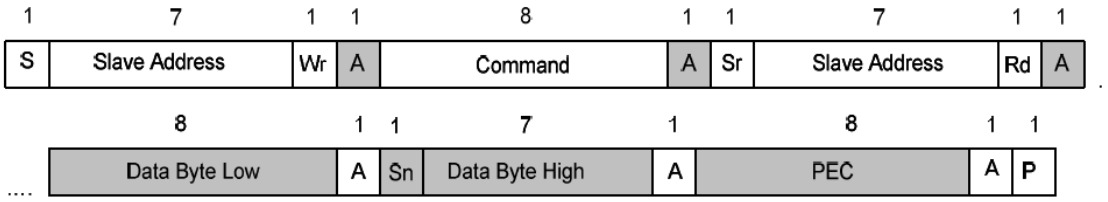


Figura 43: Protocolo de lectura 2

Como podemos observar, el formato es exactamente el mismo que el de lectura de la memoria RAM, por lo tanto también serán aproximadamente iguales los pasos a seguir. Cambiaremos las direcciones a las que quiero acceder y el comando, por ejemplo, si queremos leer la dirección del SMBus, enviaremos el comando **0b0010_1110,** puesto que accedemos a 0x0E:

1. Enviar el bit de START.
2. Enviar la dirección del esclavo (0x00*) + 1(bit de lectura).
3. Enviar comando (0b0010_1110).
4. Enviar un repetitivo START_bit.
5. Enviar la dirección del esclavo (0x00*) + 1(bit de lectura).
6. Leer el Dato Byte Low (el maestro debe devolver un bit ACK).
7. Leer el Dato Byte High (el maestro debe devolver un bit ACK).
8. Leer el PEC (el maestro puede devolver un ACK o NACK)
9. Enviar el bit STOP.

Para acceder a las diferentes direcciones de la EEPROM los comandos que debemos utilizar son los siguientes:

- 000h - **0b0010_0000**
- 001h - **0b0010_0001**
- 002h - **0b0010_0010**
- 003h - **0b0010_0011**
- 004h - **0b0010_0100**
- 005h - **0b0010_0101**
- 00Eh - **0b0010_1110**

El esquema de bloques que representaría un proceso de lectura tanto para la memoria RAM como para la memoria EEPROM sería el representado en la Figura 44.

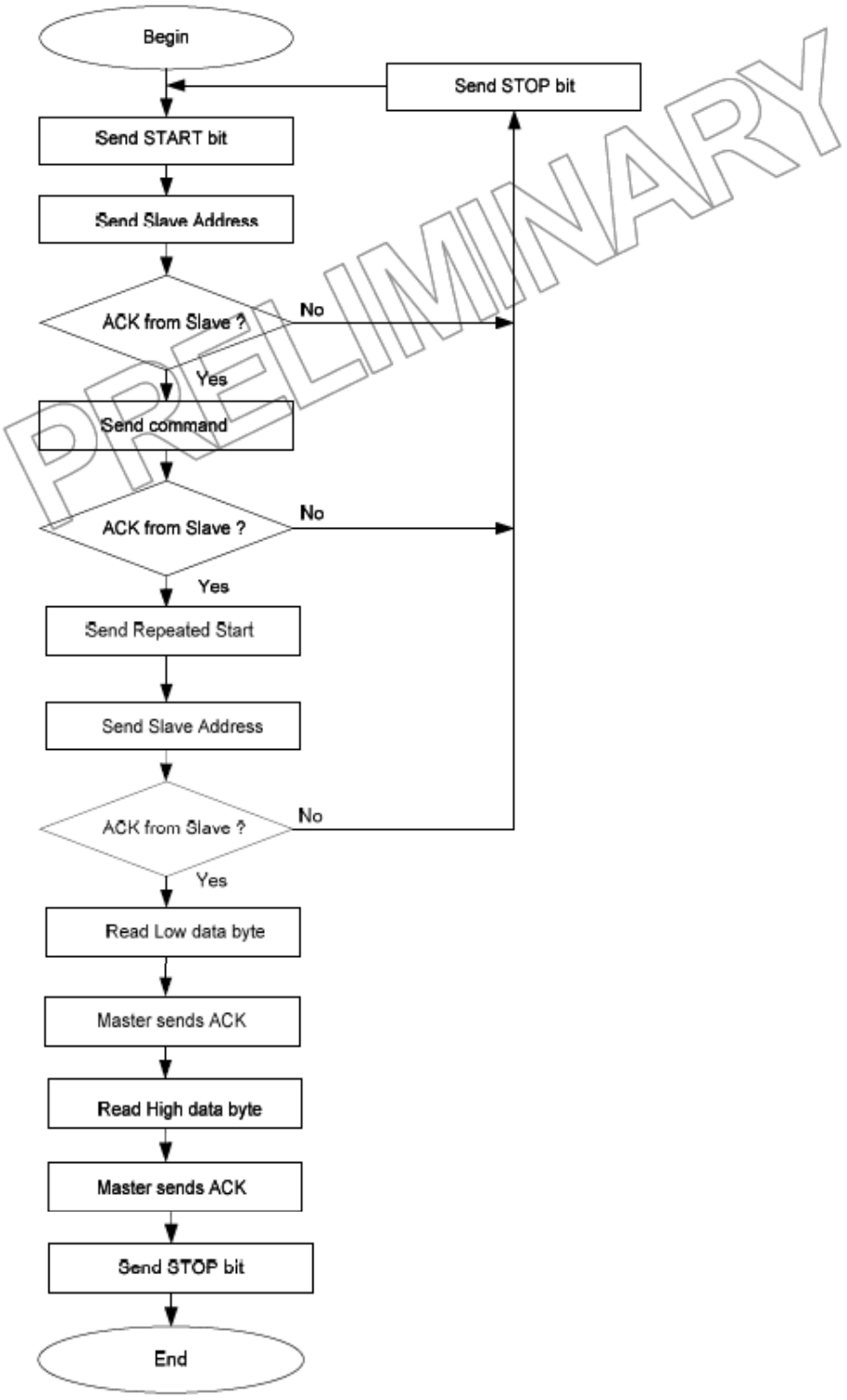


Figura 44: Diagrama de bloques de lectura de memoria RAM y EEPROM

- *Escritura en la EEPROM*

Las celdas de la memoria EEPROM que pueden ser escritas para la configuración del sensor por parte del usuario son las siguientes:

- 000h - corresponde a la Tomax (temperatura del objeto máxima)
- 001h - corresponde a la Tomin(temperatura del objeto mínima)
- 002h - corresponde al PWMCTRL (control de la salida digital)
- 003h - corresponde a Ta range (rango de la temperatura ambiente)
- 004h - corresponde al coeficiente de corrección de la emisividad del objeto.
- 005h - corresponde al Config Register1 (registro de configuración)
- 00Eh - corresponde a la dirección del SMBus (dirección del esclavo)

Intentar escribir una celda que no permite este tipo de accesos no produce ningún cambio.

La dirección 002h de PWMCTRL consiste en bits de control para la configuración de los pines de la salida PWM/SDA que se resumen en la Tabla 8 siguiente pero que en nuestro prototipo no utilizaremos:

Bit 0	Select the type of PWM mode:	1 - Single PWM, factory default for MLX90614xAx	0 – Extended PWM, factory default for MLX90614xBx
Bit 1	Enable/disable the PWM:	1 - Enable PWM, disable SMBus	0 - Disable PWM (Enable SMBus), Factory default
Bit 2	Configuration of the pin PWM:	1 - Push-Pull,	0 – Open Drain NMOS, factory default
Bit 3	Mode selection	1 - Thermo Relay,	0 - PWM, Factory default
Bits[8:4]	Extended PWM configuration	Number of repetitions divided by 2 of sensor 1 and 2 in Extended PWM mode. The number of repetitions can vary from 0 to 62 times.	
Bits[15:9]	PWM period	PWM period 1.024* ms (Single PWM) or 2.048* ms (Extended PWM) multiplied by the number written in this place. (128 in case the number is 0.)	

Tabla 8: Bits de control del registro PWM del sensor

Ocurre lo mismo con la dirección 005h que corresponde con el *ConfigRegiste1 (*Registro de configuración 1*)* que consiste en bits de control para la configuración de las partes analógicas y digitales y que en nuestro prototipo no utilizaremos, además, el fabricante recomienda no hacerlo.

Tan sólo se exponen las tablas de manera ilustrativa para describir la esctructura de la memoria y los registros del sensor pero estos bits y registros no deben ser alterados (excepto en casos especiales) puesto que se encargan de mantener la calibración de serie importante, sobre todo *Ke [15..0] ; ConfigRegister1 [13..11;7;3] ; direcciones 00Fh y 019h.*

<table>
<tr><td>Bits[2:0]</td><td>– Configure coefficients of IIR digital filter:</td><td>
<table>
<tr><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th><th>a_1</th><th>b_1</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0.5</td><td>0.5</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0.25</td><td>0.75</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0.167</td><td>0.833</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0.125</td><td>0.875</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0 (IIR bypassed)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0.8</td><td>0.2</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0.67</td><td>0.33</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0.57</td><td>0.43</td></tr>
</table>
</td></tr>
<tr><td>Bit 3</td><td>– Configure the type of ambient temperature sensor:</td><td>1 - PTC | 0 – PTAT</td></tr>
<tr><td>Bits[5:4]</td><td>– Configure the type of data transmitted through PWM:</td><td>
<table>
<tr><th>Bit 5</th><th>Bit 4</th><th>Data 1</th><th>Data 2</th></tr>
<tr><td>0</td><td>0</td><td>Ta</td><td>IR 1</td></tr>
<tr><td>0</td><td>1</td><td>Ta</td><td>IR 2</td></tr>
<tr><td>1</td><td>1</td><td>IR 1</td><td>IR 2</td></tr>
<tr><td>1</td><td>0</td><td>IR 2</td><td>Undefined*</td></tr>
</table>
</td></tr>
<tr><td>Bit 6</td><td>– Define the number of IR sensors:</td><td>1 – 2 sensors | 0 -1 sensor</td></tr>
<tr><td>Bit 7</td><td>– Define the sign Ks (Ks=dAlpha/dTobj) :</td><td>Factory calibration, do not alter</td></tr>
<tr><td>Bits[10:8]</td><td>– Configure coefficient N of FIR digital filter:</td><td>
<table>
<tr><th>Bit 10</th><th>Bit 9</th><th>Bit 8</th><th>N</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>8**</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>16**</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>32**</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>64**</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>128</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>256</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>512</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1024</td></tr>
</table>
</td></tr>
<tr><td>Bits[13:11]</td><td>Gain of IR input amplifier</td><td>Factory calibration, do not alter</td></tr>
<tr><td>Bits[15:14]</td><td>Factory calibration, do not alter</td><td></td></tr>
</table>

Tabla 9: Bits del registro ConfigRegister1 del sensor

Como ya hemos comentado anteriormente, todos los dispositivos MLX90614 responderán a la dirección de esclavo 0x00, pero puesto que en un bus de comunicación serie I2C puede haber hasta 127 dispositivos, varios de los cuales podrían ser sensores MLX90614, es conveniente dar a cada uno de los sensores una dirección de esclavo diferente. Para ello debemos reconfigurar (escribir) la celda SMBus, cuya dirección es 0x0E.

Antes de escribir en la EEPROM debe realizarse la operación de borrado. Una operación de borrado es simplemente escribir ceros en las celdas correspondientes de la EEPROM. Después del borrado o la escritura es necesaria una espera de 5ms para que la celda adopte el nuevo valor. Después de la escritura es muy recomendable que el dispositivo se reinicie encendiendo o apagando la fuente de energía o haciendo un in/out del modo sleep.

El diagrama de bloques para el borrado/escritura de la EEPROM sería el representado en la Figura 45.

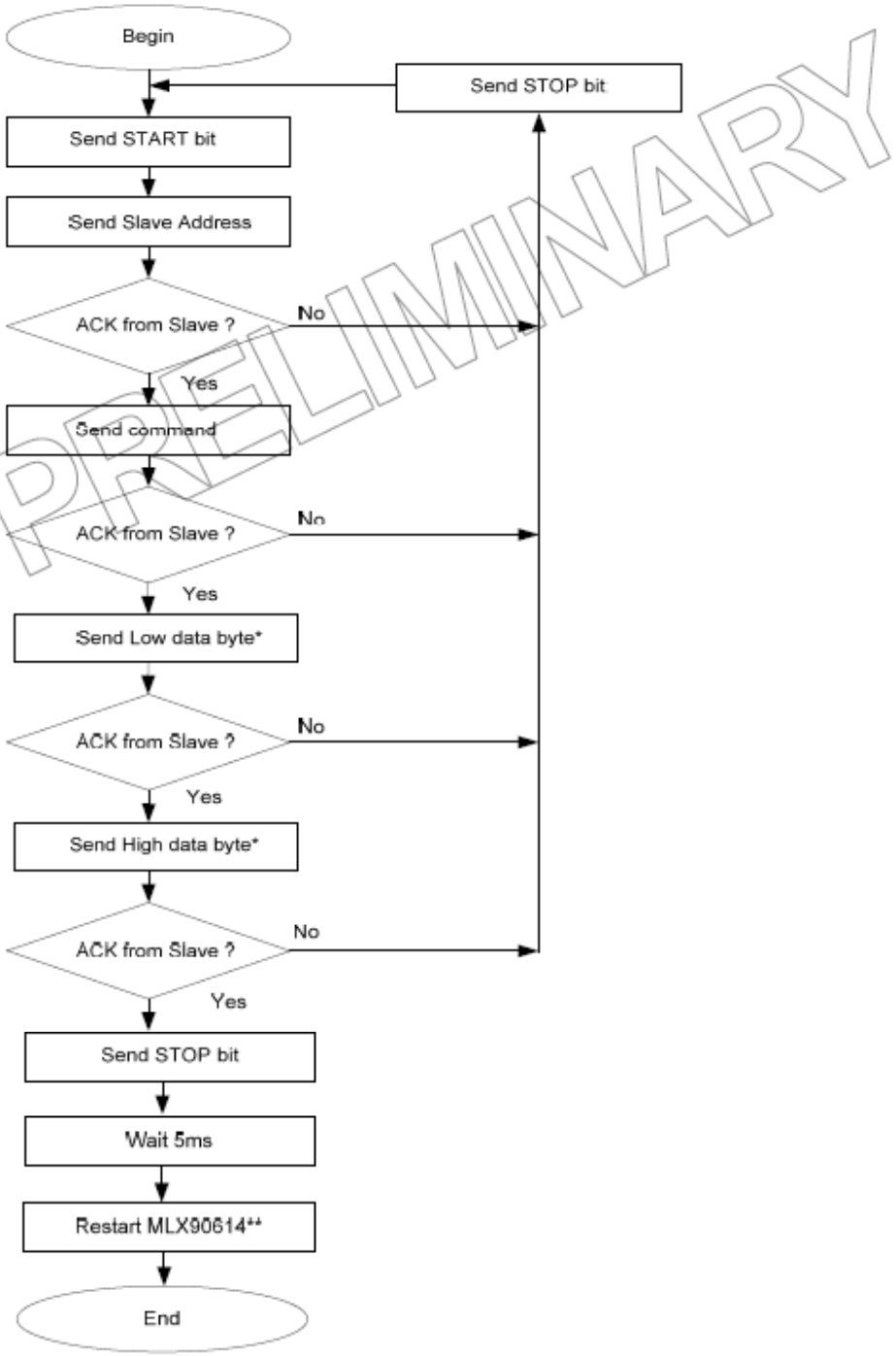


Figura 45: Diagrama de bloques de las funciones que escriben en la EEPROM

Descripción del software desarrollado

El código implementado en realidad son dos, un código principal que contiene los prototipos de las funciones, las llamadas a las funciones y los

diferentes buffers que utilizaremos para la escritura y la lectura de los registros; y otro código llamado I2C_lib en el cual se describen las diferentes funciones.

Código principal

El código principal comienza con los prototipos de las funciones a las que posteriormente iremos llamando para realizar la configuración del sensor y para las lecturas correspondientes, pero antes de los prototipos se declara la llamada a la librería LPC2100.h que incluye toda una serie de instrucciones para el funcionamiento del microprocesador. Este archivo ha sido desarrollado por el mismo fabricante, Keil, para poder manejar las herramientas del microprocesador LPC21xx de ARM7.

Tras los prototipos de las funciones que posteriormente describiremos, se declaran varios buffers para el almacenamiento de los datos con los que configuraremos el sensor. Estos buffer son:

- bufferRead
- bufferWriteSMBus
- bufferWriteTomax
- bufferWriteTomin
- bufferWriteTarange
- bufferWriteemisividad
- bufferErase

Todos los buffers son de 3 bytes dado que nosotros nos centraremos en la configuración de la EEPROM del sensor, y las direcciones de la misma son de 24 bits esto se debe a que los datos que se mandan al sensor tienen la estructura LowData, HighData y PEC, es decir, bits menos significativos de los datos, bits más significativos y código de error. El código del error no lo calcularemos ni lo utilizaremos pero sí le daremos un valor predefinido para que el código funcione y no presente fallos.

Se ha decidido almacenar previamente los datos a comunicar en buffers, porque el registro en el cual iremos colocando los datos para su posterior envío tan sólo es de un byte.

bufferRead: Espacio destinado a almacenar los datos de los registros del sensor que leeremos. Estos cambiarán según el registro que leamos.

```
char bufferRead[3];    // espacio para almacenar los bytes a leer
```

Figura 46: buffer de lectura

bufferWriteSMBus: En este buffer almacenaremos la dirección nueva que queremos dar al sensor como esclavo. Inicialmente todos los sensores responderán a la llamada por la dirección 0x00 pero como nosotros tenemos más de un dispositivo conectado al bus I2C, el sensor y el controlador del motor, daremos a cada uno una dirección diferente, para comunicarnos con uno u otro.

```
bufferWriteSMBus[0]=0x5A;    // nueva dirección del esclavo
bufferWriteSMBus[1]=0x00;    // HighData
bufferWriteSMBus[2]=0xE1;    // PEC
```

Figura 47: buffer para escribir la dirección del esclavo

bufferWriteTomax: Este buffer almacena los datos para configurar el registro Tomax de la EEPROM del sensor, es decir, la temperatura del objeto máxima. Esta temperatura se ha calculado, de acuerdo con la hoja de características del sensor (Anexos) es decir, a 125º.

```
bufferWriteTomax[0]=0x74;     // LowData
bufferWriteTomax[1]=0x63;     // HighData
bufferWriteTomax[2]=0xE1;    // PEC
```

Figura 48: buffer para la temperatura del objeto máxima

bufferWriteTomin: Almacenará los datos de configuración correspondientes a la temperatura mínima del objeto. Esta temperatura será de -40º según la hoja de características del sensor.

```
bufferWriteTomin[0]=0x22d;    // LowData
bufferWriteTomin[1]=0x0B;     // HighData
bufferWriteTomin[2]=0xE1;    // PEC
```

Figura 49: buffer para la temperatura del objeto mínima

bufferWriteTarange: Almacenará los datos de Almacenará los datos de configuración correspondientes al rango de temperaturas del ambiente. Esta temperatura será de 125º según la hoja de características del sensor.

```
bufferWriteTarange[0]=0x74;   // LowData
bufferWriteTarange[1]=0x63;   // HighData
bufferWriteTarange[2]=0xE1;    // PEC
```

Figura 50: buffer para el rango de temperatura ambiente

bufferWriteemisividad: La emisividad es la proporción de radiación térmica emitida por una superficie u objeto debida a una diferencia de temperatura determinada. Es la transferencia de calor debida a la emisión de ondas electromagnéticas entre dos superficies y no necesita de un medio para llevarse a cabo. Para nuestro sensor, estará configurada con un factor de 1.0 (según hojas de características) [datasheet MLX90614]

```
bufferWriteemisividad[0]=0xFF;  // LowData
bufferWriteemisividad[1]=0xFF;  // HighData
bufferWriteemisividad[2]=0xE1;   // PEC
```

Figura 51: buffer para la emisividad del objeto

bufferErase: Puesto que previamente ha realizar la escritura de alguno de los registros de la memoria EEPROM del sensor se ha de realizar un borrado de los mismo, debemos crear un buffer de ceros, es decir, el borrado será la escritura de un 0x00 en el registro correspondiente.

```
bufferErase[0]=0x00;    //LowData
bufferErase[1]=0x00;    //HighData
bufferErase[2]=0x6F;    //Código de error
```

Figura 52: buffer de borrado.

Como se puede deducir, para configurar el sensor con otros valores de temperatura, emisividad u otra dirección del esclavo, no debemos modificar las funciones, tan sólo, y de un modo muy sencillo, modificaremos los datos almacenados en los buffers.

A las funciones propiamente dichas, además de enviarles los datos que deben escribir en los registros hemos de enviarles las direcciones en la EEPROM de esos registros. Estas direcciones siempre serán las mismas, por lo tanto, las definiremos para no tener que recordar su valor hexadecimal.

```
#define Tobj1RAM  0x07  //temperatura objeto2
#define Tobj2RAM 0x08; //temperatura objeto1
#define TaRAM 0x06;    //temperatura ambiente
#define SMBusEEPROM 0x0E;  //dirección del esclavo
#define TomaxEEPROM 0x40; //temperatura del objeto máxima
#define TominEEPROM 0x41; //temperatura del objeto mínima
#define PWMctrlEEPROM 0x42; //control de la salida digital
#define TarangeEEPROM 0x43; //rango de la temperatura ambiente
#define emisividadEEPROM 0x44; //coeficiente de corrección de emisividad
#define configregister1EEPROM 0x45; //registro de configuración
```

Figura 53: direcciones de memoria

Como podemos observar en la Figura 53, existen registros a utilizar tanto en la memoria RAM como en la EEPROM, pero como se ha comentado anteriormente, los registros RAM no pueden ser modificados, tan sólo leídos.

Tras todas estas definiciones, procederemos a llamar a las funciones, primero haremos una llamada a la función borrado (*EraseEEPROM_I2C*) para que el registro correspondiente quede listo para la escritura (*WriteEEPROM_I2C*) y posteriormente grabaremos los datos de la configuración con la función escritura. Un ejemplo de esta llamada a la función sería el indicado en la Figura 54:

```
EraseEEPROM_I2C(0x00, SMBusEEPROM, bufferErase);
WriteEEPROM_I2C(0x00, SMBusEEPROM, bufferWriteSMBus);
```

Figura 54: Llamada a la función de modificación de la dirección del bus

En la llamada se puede observar que primero enviamos (a ambas funciones) la dirección genérica del esclavo 0x00, puesto que aún no la hemos cambiado, el registro que queremos modificar, SMBusEEPROM y por último el buffer que almacena los datos a escribir.

Código I2C_lib

En éste código se incluyen las definiciones de todas las funciones para realizar la comunicación, la configuración y la inicialización del sistema.

Las funciones que incluye son las siguientes:

```
void EraseEEPROM_I2C(unsigned char S_Addr, unsigned char Command, char *bufferErase);
void WriteEEPROM_I2C(unsigned char S_Addr, unsigned char Command, unsigned char *bufferWrite);
void ReadEEPROM_I2C(unsigned char S_Addr, unsigned int NumBytes, char *bufferRead, unsigned char Command);
void Init_I2C(unsigned char SCLL, unsigned char SCLH);
void SleepMode(unsigned char S_Addr);
```

Figura 55: Prototipos de las funciones del código del sensor

Las funciones *EraseEEPROM_I2C, WriteEEPROM_I2C y ReadEEPROM_I2C* se han implementado para llevar a cabo el protocolo de comunicación I2C descrito con detalle en los apartados anteriores, esto es, esperar siempre a que se reciba el bit de confirmación de datos leídos ACK para transmitir los siguientes.

Las funciones *EraseEEPROM_I2C* y *WriteEEPROM_I2C* reciben como información del programa principal: la dirección del esclavo, es decir, la del sensor, que será 0x00 hasta que no lo cambiemos a través de la escritura del registro SMBus, lo cual realizaremos en último lugar, pero esto puede modificarse cambiando simplemente de orden las instrucciones de llamada a

las funciones, es decir, llamando la primera a la función de configuración del SMBus.

También recibirán como datos, la dirección del registro de memoria que hay que modificar y el buffer correspondiente con la información.

Hemos de puntualizar además que para que una celda de la memoria EEPROM quede reescrita se debe hacer una espera de al menos 5 milisegundos tras la escritura y además es recomendable desactivar y volver a activar el sensor.

La función *ReadEEPROM_I2C* además de la dirección del esclavo, la dirección del registro y el buffer donde se almacenarán los datos, recibe también la información del número de datos que han de ser leídos.

La función *Init_I2C* establece todos los parámetros de inicialización del interfaz I2C en el microprocesador, configura los puertos del mismo P0.2 y P0.3 para que actúen como líneas SDA y SCL y también configura la frecuencia del reloj.

Además hemos descrito otra función llamada *SleepMode* que hace que el sensor entre en un estado de baja energía o Standby.

El código que se ha desarrollado y descrito se detalla en los anexos de éste mismo documento.

3.2.2 Software del motor

El software que desarrolla el control del motor es muy similar al del sensor. La comunicación sigue el mismo patrón puesto que se trata de transmisión de información a través del mismo bus (I2C) y desde el mismo microprocesador. Las diferencias radicarán en los datos a enviar y a leer.

Puesto que este software debe controlar el motor, debe incluir funcionalidades como el ajuste de la posición, la velocidad, el modo de funcionamiento etc. Pero esto no es más que datos enviados a distintos registros del controlador, por lo tanto, podemos conservar la estructura que utilizamos para el sensor.

Descripción de la comunicación

El protocolo de comunicación es exactamente el mismo que el descrito con anterioridad para el sensor, por lo tanto no volveremos a explicarlo, tan sólo puntualizaremos algunas características. [web15]

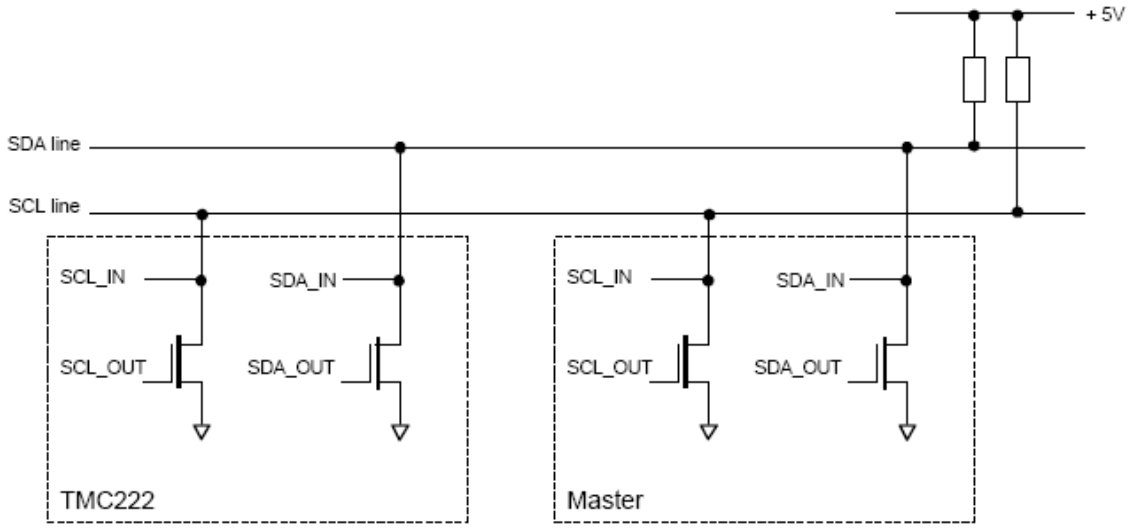


Figura 56: Interfaz de la comunicación serie I2C

Cada byte introducido en la línea SDA debe tener una longitud de 8 bits, donde el bit más significativo (MSB) se transfiere primero. El número de bits que pueden ser transmitidos al TMC222 está restringido a 8 bytes, cada byte, como en el sensor, debe estar seguido de un bit de reconocimiento (ACK). De nuevo en la Figura 57 se muestra la transmisión de datos a través del I2C.

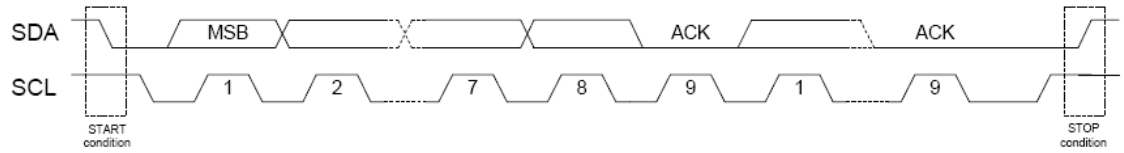


Figura 57: Transferencia de datos a través del I2C

El TMC222 debe ser provisto de una dirección de esclavo, aunque al igual que el sensor, responde a la dirección 0x00. Esto es mejor así puesto que así discriminamos los diferentes dispositivos conectados al mismo bus.

A diferencia del caso del sensor, esta dirección se codifica según 7 bits (ver Tabla 10) dos de los cuales están internamente puestos a uno (AD6 y AD5). Es una combinación por lo tanto de 4 bits de memoria OTP, que se explicará más adelante en qué consiste y un bit de cableado externo (hardwired bit).

AD6	AD5	AD4	AD3	AD2	AD1	AD0	Physical address
'1'	'1'	OTP_AD3	OTP_AD2	OTP_AD1	OTP_AD0		OTP Memory
						HW2	Hardwired Bit (Connect to 0 or 1)

Tabla 10 : Campo de dirección física del TMC222

Para nuestro caso, con una memoria OTP no programada (OTP_AD3, OTP_AD2, OTP_AD1, OTP_AD0 a nivel bajo 0) y el bit cableado puesto a 1 (lo

conectaremos a la tensión de alimentación), la dirección del esclavo predeterminada sería; para leer 0xC3 y 0xC2 para escribir.

La transferencia de datos desde el maestro al esclavo, es decir, la escritura de datos seguiría la estructura de la Figura 58.

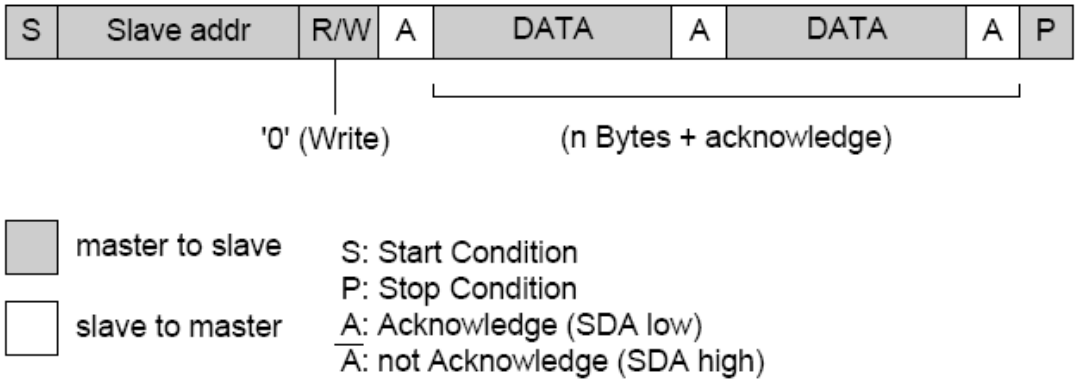


Figura 58: Escritura de datos en el esclavo

Y la lectura de datos por parte de un esclavo se ilustra en la Figura 59.

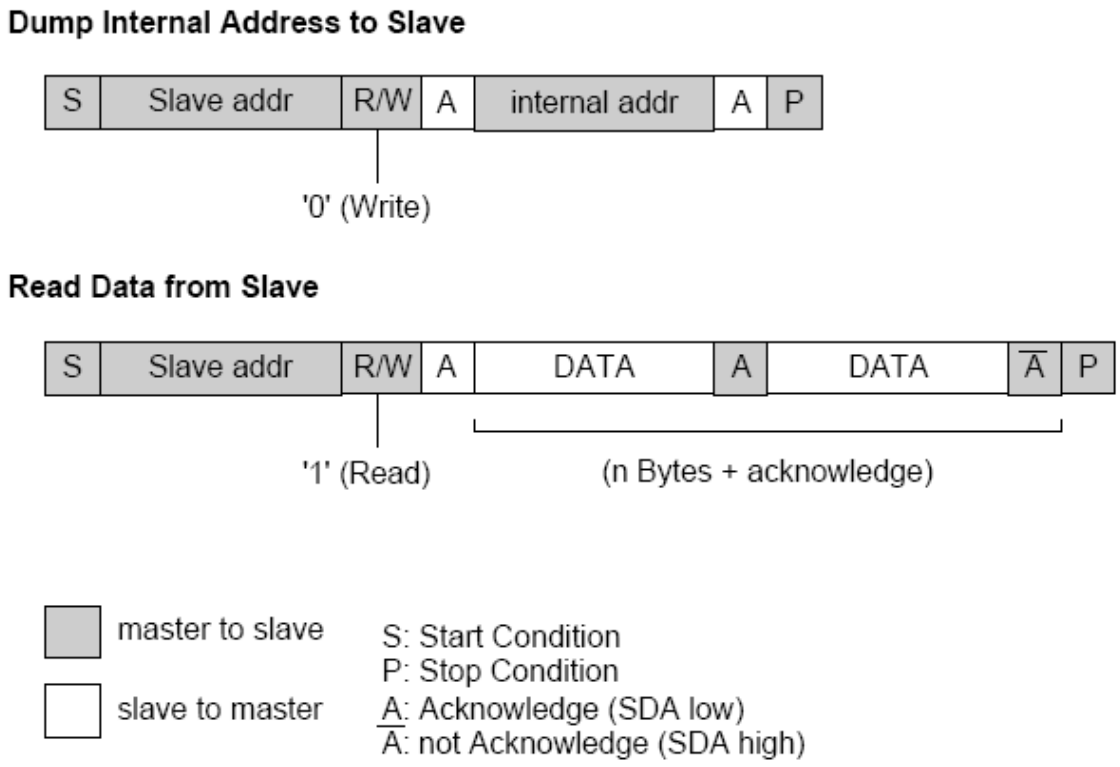


Figura 59: Lectura de datos desde el esclavo.

El primer esquema (Dump Internal Address to slave) consiste en la escritura de dos bytes, lo cual significa que estamos comunicando desde el maestro la dirección del esclavo con el cual nos queremos comunicar y la dirección interna particular que queremos leer de ese esclavo.

El segundo esquema (Read data from slave) determina la lectura de los datos, en este caso, es el esclavo el emisor de información y el maestro la recibe. El maestro en nuestro caso es el microprocesador, y el esclavo seria el controlador.

La comunicación además debe llevarse a cabo a través de ciertos comandos predeterminados los cuales se muestran en la Tabla 11.

Los comandos de lectura se utilizará para:

- Obtener información del estado actual
- Obtener la posición del motor
- Verificar la programación y configuración del TMC222

Los comandos de escritura se utilizarán para:

- Programar la memoria OTP
- Configurar el controlador con los parámetros de movimiento
- Determinar el objetivo de posición del motor.

Command Mnemonic	Function	Command Byte (hexadecimal)
GetFullStatus1	Returns complete status of the chip	0x81
GetFullStatus2	Returns actual, target and secure position	0xFC
GetOTPParam	Returns OTP parameter	0x82
GotoSecurePosition	Drives motor to secure position	0x84
HardStop	Immediate full stop	0x85
ResetPosition	Sets actual position to zero	0x86
ResetToDefault	Overwrites the chip RAM with OTP contents	0x87
RunInit	Reference Search	0x88
SetMotorParam	Sets motor parameter	0x89
SetOTPParam	Zaps the OTP memory	0x90
SetPosition	Programmers a target and secure position	0x8B
SoftStop	Motor stopping with deceleration phase	0x8F

Tabla 11: Comandos del TMC222

Estos parámetros con su función (que ahora aparece en inglés) se explicarán y detallarán en apartados posteriores.

Descripción de las funciones

Para posibilitar el movimiento del motor a una posición determinada el controlador debe realizar un determinado tipo de funciones como el control de la velocidad, de la posición y del modo de funcionamiento. [datasheet TMC222]

Modos de paso

El controlador TMC222 soporta hasta 16 micro pasos por un paso completo, lo cual conduce a un movimiento suave del motor.

El motor tiene cuatro modos de funcionamiento dependiendo de la resolución de los micro pasos y que pueden ser seleccionados por el usuario.

- Modo de medio paso
- Modo de un cuarto de paso
- Modo de un octavo de paso
- Modo de un decimosexto de paso

En las Figura 60,61,63, 64 se muestran las diferentes formas de ondas de las corrientes que alimentan al motor en los diferentes modos de funcionamiento.

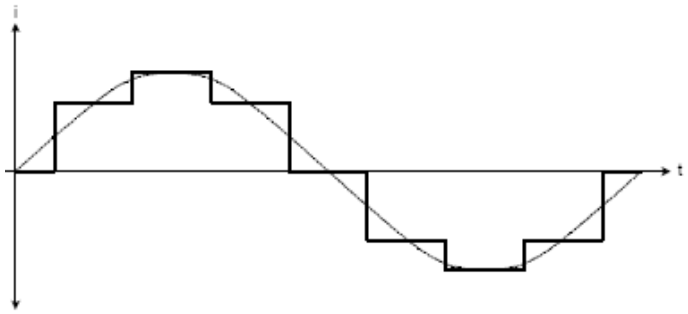


Figura 60 : Corriente para el modo ½ paso

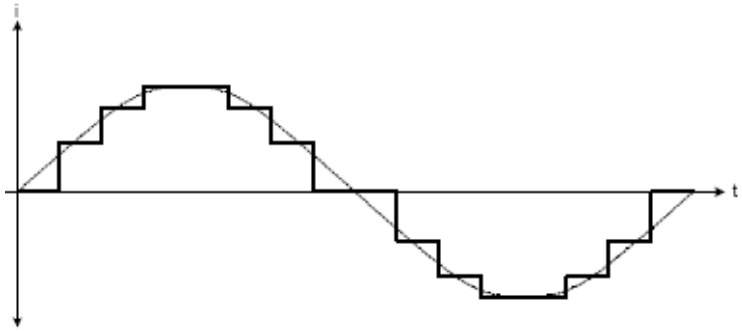


Figura 61 : Corriente para el modo ¼ de paso

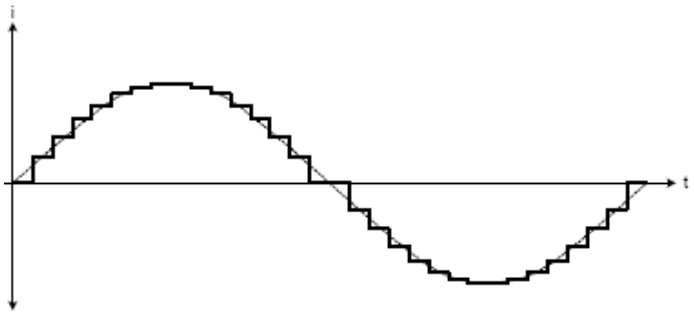


Figura 63 : Corriente para el modo 1/8 de paso

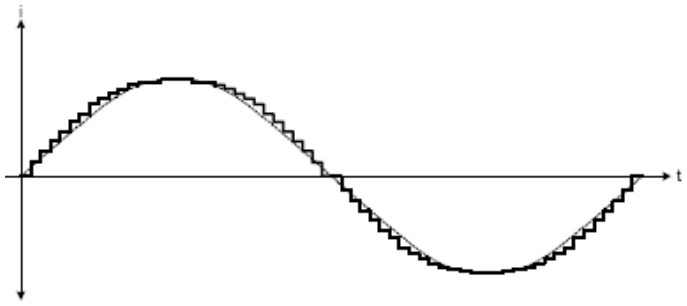


Figura 64: Corriente para el modo 1/16 de paso

Rampa de velocidad

Una rampa de velocidad común en la cual el motor se mueve hacia la posición deseada, se muestra en la Figura 65.

El movimiento consiste en una fase de aceleración, una fase de velocidad constante y una fase de deceleración.

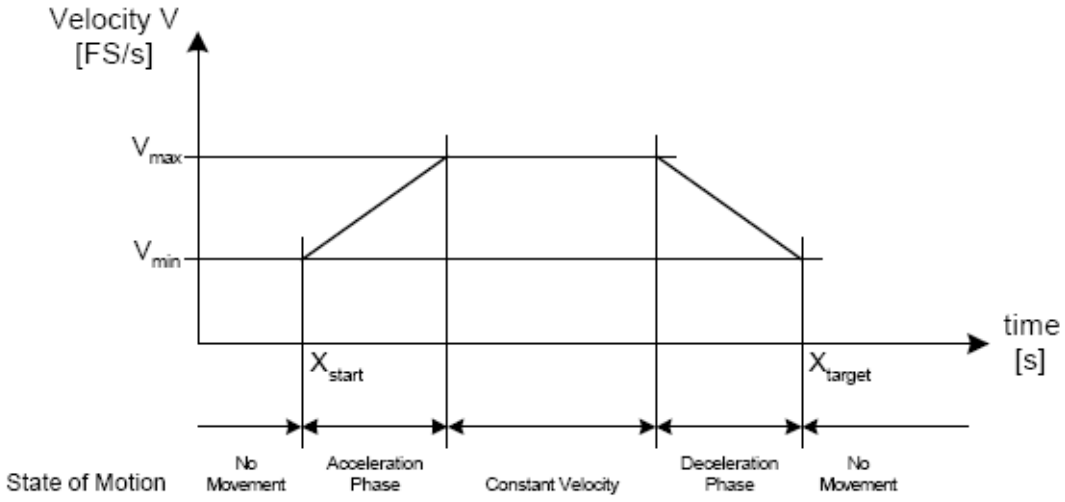


Figura 65: Movimiento básico del motor

El movimiento básico comienza (Xstart) con una velocidad mínima de comienzo (Vmin). Durante la fase de aceleración (acceleration phase) la velocidad se incrementa hasta una alcanzar una velocidad máxima (Vmax). Tras la fase de aceleración, el movimiento continúa a la velocidad máxima hasta que ha de ser decrementada para parar en la posición deseada.

Ambos parámetros de velocidad Vmax y Vmin, al igual que el factor de aceleración, pueden ser programables. Vmin siempre será un ratio de Vmax.

El número de pasos enteros equivalentes durante la fase de aceleración puede ser calculada con la siguiente ecuación:

$$N_{pasos} = \frac{V_{max}{}^2 - V_{min}{}^2}{2 * Acc}$$

Además se ha de tener en cuenta que Vmin no puede ser modificada mientras se desarrolla un movimiento y Vmax sólo permite cambios en circunstancias especiales.

El valor pico de corriente para alimenta a cada bobina del motor paso a paso se puede seleccionar de la Tabla 11 entre 16 posibles valores. Hay que distinguir entre dos corrientes, Irun e Ihold. Irun alimenta el motor paso a paso mientras un movimiento se lleva a cabo, sin embargo, Ihold es la corriente que se necesita para mantener el motor antes o después de un movimiento, en la Figura 66 se ilustra el mecanismo de funcionamiento al final del movimiento del motor.

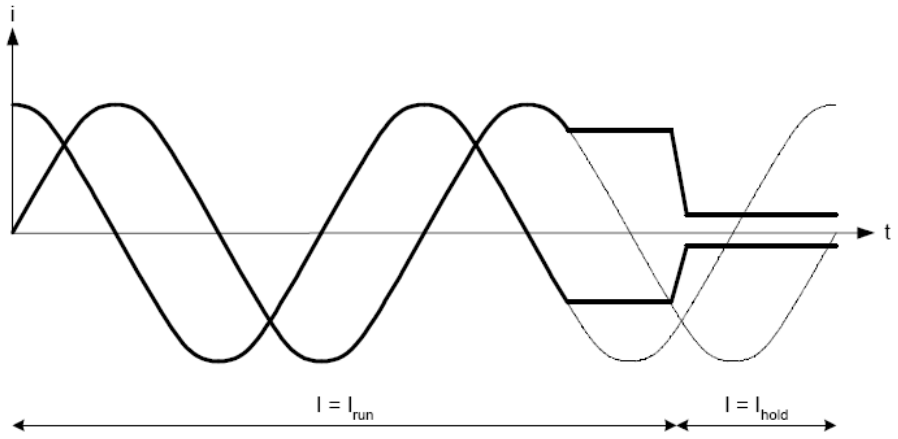


Figura 66: Mecanismo de transición entre corrientes

En las figuras a continuación mostraremos diferentes ejemplos de rampas de velocidad.

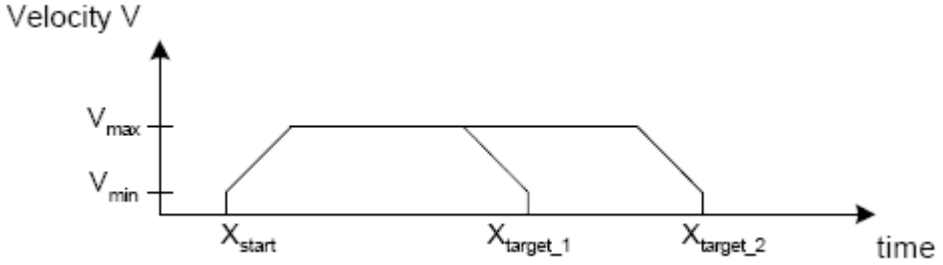


Figura 67: Movimiento con cambio de posición final

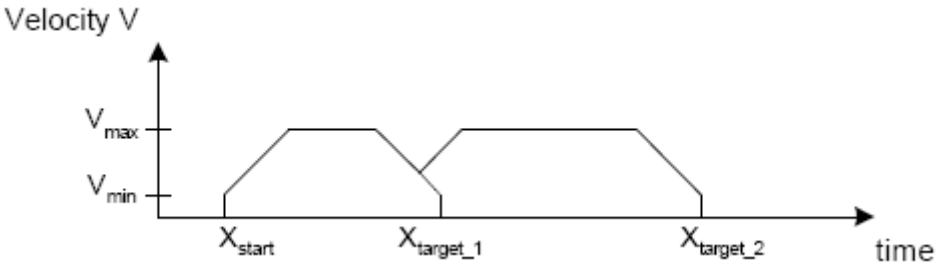


Figura 68: Movimiento con cambio de posición final en la fase de deceleración

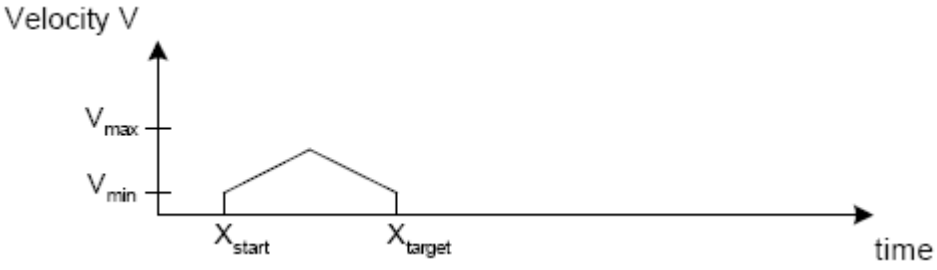


Figura 69: Movimiento corto, no se llega a la Vmax

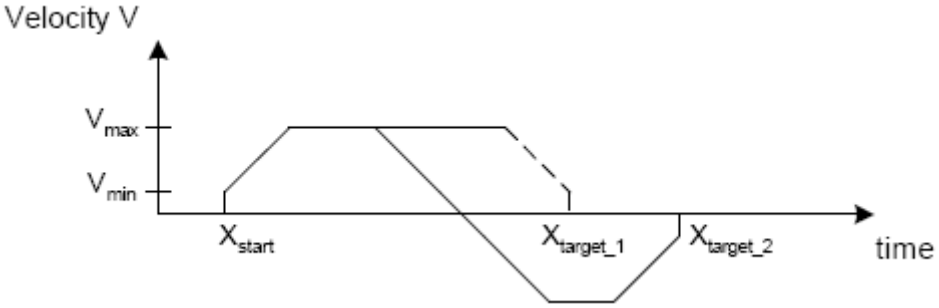


Figura 70: Cambio de posición final en la dirección opuesta

Rangos de posición

La información de la posición está codificada utilizando el formato en complemento a dos. Dependiendo del modo del motor de paso a paso los rangos de posición son los que se indican en la Tabla 12. En ella se indican el modo de paso (Stepping mode), el rango de posición (position range) y el rango total de ejecución.

Las posiciones finales pueden ser programadas mediante el comando SetPosition que veremos más adelante.

La posición actual puede ser leída mediante el comando GetFullStatus2 que también detallaremos más adelante.

Stepping Mode	Position Range	Full range excursion
Half-stepping	-4096…+4095 $(-2^{12}…+2^{12}-1)$	8192 half-steps 2^{13}
1/4 micro-stepping	-8192…+8191 $(-2^{13}…+2^{13}-1)$	16384 micro-steps 2^{14}
1/8 micro-stepping	-16384…+16383 $(-2^{14}…+2^{14}-1)$	32768 micro-steps 2^{15}
1/16 micro-stepping	-32768…+32767 $(-2^{15}…+2^{15}-1)$	65536 micro-steps 2^{16}

Tabla 12: Rangos de posición

Además existe una posición de seguridad pre-programable mediante el comando GotoSecurePosition. Sin embargo la resolución es menor que para los comandos de posición normal.

Es importante saber que el motor de pasos no proporciona información sobre la posición actual del motor. Por lo tanto, es recomendable habilitar una posición de referencia en caso de que el motor se apague.

Manejo de los modos de apagado

El motor, para autoprotegerse, entrará en el modo de apagado cuando ocurra alguna de las siguientes situaciones:

- La temperatura del dispositivo se eleva por encima del umbral permitido.
- El voltaje se desploma por debajo de un valor predeterminado
- Ocurre algún problema eléctrico
- Falla el surtidor de carga

Durante el apagado del motor se desarrollan las siguientes funciones desde el controlador principal:

- La posición final registrada se carga en los registros de posición actual.

La comunicación serie se mantiene activa durante el apagado del motor. Para abandonar el estado de apagado las condiciones siguientes deben ser verdaderas:

- Las condiciones que llevaron al motor a su apagado no están activas ya.
- Se lleva a cabo el comando GetFullStatus1 a través de la comunicación serie.

Memorias Ram y OTP

Algunos de los registros de la memoria RAM son inicializados con el contenido de la memoria OTP (One Time Programmable, programable una vez), sin embargo, éstos pueden ser cambiados posteriormente a la inicialización.

En la Tabla 13 se muestran los diferentes registros de la memoria RAM del controlador.

Todos los registros, los comandos asociados y los comentarios serán detallados más adelante en esta memoria.

Register	Mnemonic	Length (bit)	Related commands	Comment	Reset State
Actual Position	ActPos	16	GetFullStatus2 ResetPosition	Actual Position of the Stepper Motor. 16-bit signed	0x0000
Target Position	TagPos	16	SetPosition GetFullStatus2 ResetPosition	Target Position of the Stepper Motor. 16-bit signed	
Acceleration Shape	AccShape	1	GetFullStatus1 SetMotorParam ResetToDefault	0 = Acceleration with Acc Parameter. 1 = Velocity set to Vmin, without acceleration	
Coil Peak Current	Irun	4	GetFullStatus1 SetMotorParam ResetToDefault	Coil current when motion is ongoing (Table 12: Irun / Ihold Settings)	OTP Memory
Coil Hold Current	Ihold	4	GetFullStatus1 SetMotorParam ResetToDefault	Coil current when motor stands still (Table 12: Irun / Ihold Settings)	
Minimum Velocity	Vmin	4	GetFullStatus1 SetMotorParam ResetToDefault	Start Velocity of the stepper motor (Table 4: Vmin)	
Maximum Velocity	Vmax	4	GetFullStatus1 SetMotorParam ResetToDefault	Target Velocity of the stepper motor (Table 3: Vmax Parameter)	
Shaft	Shaft	1	GetFullStatus1 SetMotorParam ResetToDefault	Direction of motion	
Acceleration / Deceleration	Acc	4	GetFullStatus1 SetMotorParam ResetToDefault	Parameter for acceleration (Table 5: Acc Parameter)	
Secure Position	SecPos	11	GetFullStatus2 ResetToDefault	Target Position for GotoSecurePosition command (6.8.4 GotoSecurePosition); 11 MSBs of 16-bit position (LSBs fixed to '0')	
Stepping Mode	StepMode	2	GetFullStatus1 GetFullStatus2 ResetToDefault	Micro stepping mode (5.1.1 Stepping Modes)	

Tabla 13 : Registros RAM del controlador TMC222

La estructura de la memoria OTP se muestra en la Tabla 14.

OTP Address	OTP Bit Order							
	7	6	5	4	3	2	1	0
0x00	OSC3	OSC2	OSC1	OSC0	IREF3	IREF2	IREF1	IREF0
0x01		TSD2	TSD1	TSD0	BG3	BG2	BG1	BG0
0x02					AD3	AD2	AD1	AD0
0x03	Irun3	Irun2	Irun1	Irun0	Ihold3	Ihold2	Ihold1	Ihold0
0x04	Vmax3	Vmax2	Vmax1	Vmax0	Vmin3	Vmin2	Vmin1	Vmin0
0x05	SecPos10.	SecPos9	SecPos8	Shaft	Acc3	Acc2	Acc1	Acc0
0x06	SecPos7	SecPos6	SecPos5	SecPos4	SecPos3	SecPos2	SecPos1	SecPos0
0x07					StepMode1	StepMode0	LOCKBT	LOCKBG

Tabla 14 : Estructura de la memoria OTP

Los parámetros alojados en las direcciones 0x00 y 0x01, así como el bit LOCKBT han sido programados por el fabricante y corresponden a la calibración del dispositivo, por lo tanto no pueden ser modificados.

El comando utilizado para cargar los parámetros de la memoria OTP en la memoria RAM vía comunicación serie I2C es el SetMotorParam.

Como podemos observar en la estructura de la Tabla 14, las direcciones útiles para realizar la configuración del TMC222 en nuestro caso son las siguientes:

- 0x03 Irun e Ihold
- 0x04 Vmax y Vmin
- 0x05 SecPos y Acc
- 0x06 SecPos

Descripción de los comandos

Los comandos de aplicación que rigen el funcionamiento del controlador y que hemos de configurar según la posición, velocidad y demás parámetros, son los que se muestran en la Tabla 15.

Command Mnemonic	Function	Command Byte (hexadecimal)
GetFullStatus1	Returns complete status of the chip	0x81
GetFullStatus2	Returns actual, target and secure position	0xFC
GetOTPParam	Returns OTP parameter	0x82
GotoSecurePosition	Drives motor to secure position	0x84
HardStop	Immediate full stop	0x85
ResetPosition	Sets actual position to zero	0x86
ResetToDefault	Overwrites the chip RAM with OTP contents	0x87
RunInit	Reference Search	0x88
SetMotorParam	Sets motor parameter	0x89
SetOTPParam	Zaps the OTP memory	0x90
SetPosition	Programmers a target and secure position	0x8B
SoftStop	Motor stopping with deceleration phase	0x8F

Tabla 15: Comandos de manejo del TMC222

A continuación procederemos a explicar la función de cada uno y cómo se implementarían en el software desarrollado aunque en nuestro dispositivo tan sólo utilizaremos algunos de ellos.

GetFullStatus1

Este comando es enviado al esclavo desde el microcontrolador para obtener información del estado completo del circuito y del motor paso a paso.

El microprocesador enviará al controlador las instrucciones mostradas en la Tabla 16 para implementar este comando.

GetFullStatus1 command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	GetFullStatus1	1	0	0	0	0	0	0	1

Tabla 16: Envío para el GetFullStatus1

Mandará primero la dirección del esclavo y más tarde el comando 0x81 (10000001) y con ello el controlador entenderá que debe responder con toda la información referente a su estado, es decir, con los parámetros mostrados en la Tabla 17.

GetFullStatus1 command (Response)									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	1
1	Address	0	0	0	OTP3	OTP2	OTP1	OTP0	HW
2	Irun & Ihold	Irun (3:0)				Ihold (3:0)			
3	Vmax & Vmin	Vmax (3:0)				Vmin (3:0)			
4	Status 1	AccShape	StepMode(1:0)		Shaft	ACC(3:0)			
5	Status 2	VddReset	StepLoss	ElDef	UV2	TSD	TW	Tinfo(1:0)	
6	Status 3	Motion(2:0)			ESW	OVC1	OVC2	1	CPFail
7	N/A	1	1	1	1	1	1	1	1
8	N/A	1	1	1	1	1	1	1	1

Tabla 17: Respuesta al comando GetFullStatus1

- Irun e Ihold: Corrientes de establecimiento y mantenimiento
- Vmax y Vmin: Velocidad máxima y mínima del motor
- Shaft: Dirección de rotación del motor (horario o antihorario)
- StepMode: Modo de funcionamiento del motor
- Acc: aceleración/deceleración del motor
- AccShape: Forma de la aceleración

Y otra información referente al estado del motor:

- Motion: estado del movimiento
- ESW: estado del interruptor externo
- VddReset: Reestablecimiento de la fuente de alimentación
- TSD: umbral de temperatura para el apagado
- TW: Aviso de temperatura
- UV2: umbral de voltaje
- EIDef: defecto eléctrico
- OVC1 y OVC2: detección de sobrecorrientes

Y otras. En nuestro dispositivo este comando no se utilizará.

GetFullStatus2

Este comando de lectura del controlador se utiliza para obtener la posición actual del motor, la posición final del movimiento y la posición de seguridad.

De nuevo el microprocesador enviará al controlador la llamada y el TMC222 responderá con la información solicitada.

GetFullStatus2 command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	GetFullStatus2	1	1	1	1	1	1	0	0

Tabla 18: Llamada del comando GetFullStatus2

GetFullStatus2 command (Response)									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	1
1	Address	0	0	0	OTP3	OTP2	OTP1	OTP0	HW
2	Actual Position 1	ActPos(15:8)							
3	Actual Position 2	ActPos(7:0)							
4	Target Position 1	TagPos(15:8)							
5	Target Position 2	TagPos(7:0)							
6	Secure Position	SecPos(7:0)							
7	Secure Position	1	1	1	1	1	SecPos(10:8)		
8	N/A	1	1	1	1	1	1	1	1

Tabla 19: Respuesta al comando GetFullStatus2

GetOTPParam

Utilizado para la obtención del contenido de la memoria OTP cuyos registros ya hemos comentado con anterioridad.

GetOTPParam command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	GetOTPParam	1	0	0	0	0	0	1	0

Tabla 20: Llamada del comando GetOTPParam

GetOTPParam command (Response)									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	1
1	OTP byte 0	OTP@0x00							
2	OTP byte 1	OTP@0x01							
3	OTP byte 2	OTP@0x02							
4	OTP byte 3	OTP@0x03							
5	OTP byte 4	OTP@0x04							
6	OTP byte 5	OTP@0x05							
7	OTP byte 6	OTP@0x06							
8	OTP byte 7	OTP@0x07							

Tabla 21: Respuesta al domando GetOTPParam

GotoSecurePosition

Comando de escritura para que el motor se mueva a la posición de seguridad. Es más bien, una instrucción que el microprocesador (master) manda al controlador TMC222 (esclavo), para que realice la acción de ir a la posición de seguridad.

GotoSecurePosition command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	GotoSecurePosition	1	0	0	0	0	1	0	0

Tabla 22: Instrucción de GotoSecurePosition

HardStop

Comando de escritura internamente provocado cuando un problema eléctrico es detectado aunque también puede ser enviado desde el microprocesador por otras razones de seguridad.

HardStop command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	HardStop	1	0	0	0	0	1	0	1

Tabla 23: Instrucción de HardStop

ResetPosition

Comando que restaura las posiciones actual y final, es decir, los registros ActPos y TagPos de la memoria RAM para permitir una inicialización de la posición del motor paso a paso.

ResetPosition command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	ResetPosition	1	0	0	0	0	1	1	0

Tabla 24: Instrucción de ResetPosition

ResetToDefault

Este comando restaura el esclavo con todos sus nodos hasta el estado inicial, por lo tanto, recargará la memoria RAM con los parámetros de restauración registrados.

El registro de posición actual ActPos no se modifica con este comando y su valor se copia al registro de posición final TagPos para evitar que el motor se posicione en 0.

ResetToDefault command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	ResetToDefault	1	0	0	0	0	1	1	1

Tabla 25: Instrucción de ResetToDefault

RunInit

Este comando de escritura se manda al esclavo con el fin de iniciar la colocación del motor en una búsqueda de la posición de referencia cero.

El comando inicia la búsqueda de referencia y consiste en los parámetros Vmax y Vmin e información de la posición final del primer y segundo movimento. En la Figura 71 se encuentra una secuencia explicativa.

Una unidad de referencia consiste en dos movimientos para conducir el motor desde una posición desconocida a una de cambio.

Las posiciones Pos1 y Pos2 pueden elegirse libremente.

Tras el segundo movimiento, la posición actual registrada se establece como cero y podrá almacenarse como posición de seguridad si se desea.

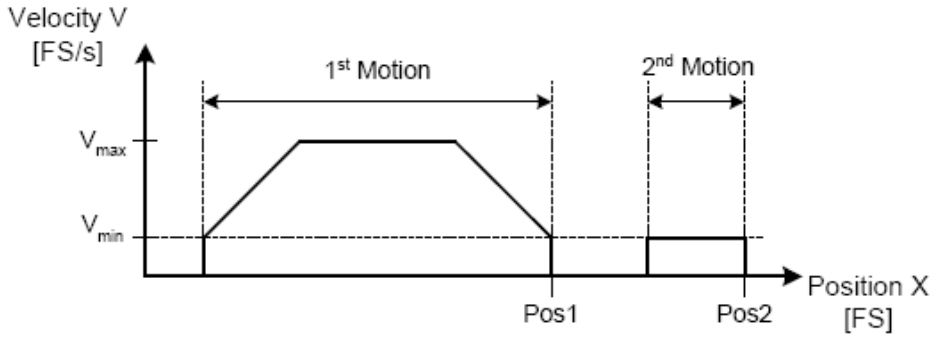


Figura 71: RunInit

Para llevar a cabo esta instrucción se llevan a cabo una serie de comandos:

- SetMotorParam (Vmax,Vmin)
- SetPosition(Pos1)
- SetMotorParam(Vmin, Vmin)
- SetPosition(Pos2)
- ResetPosition
- GotoSecurePosition

El comando RunInit no puede ser interrumpido excepto en las condiciones de apagado del motor.

El microprocesador debe asegurarse que la posición final no es igual a la posición actual del motor paso a paso y que las posiciones finales de los movimientos primero y segundo también son diferentes.

RunInit command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	RunInit	1	0	0	0	1	0	0	0
2	N/A	1	1	1	1	1	1	1	1
3	N/A	1	1	1	1	1	1	1	1
4	Vmax Vmin	Vmax(3:0)				Vmin(3:0)			
5	Position1 byte 1	TagPos1(15:8)							
6	Position1 byte 2	TagPos1(7:0)							
7	Position2 byte 1	TagPos2(15:8)							
8	Position2 byte 2	TagPos2(7:0)							

Tabla 26: Instrucción RunInit

SetMotorParam

Comando de escritura para establecer los valores de los siguientes parámetros de los registros de la memoria RAM:

- Corriente máxima de pico Irun
- Corriente de mantenimiento Ihold
- Velocidad máxima para el motor Vmax
- Velocidad mínima para el motor Vmin
- Forma de la aceleración AccShape
- Modo de paso del motor StepMode
- Dirección de giro del motor Shaft
- Aceleración del motor Acc
- Posición de seguridad SecPos

SetMotorParam command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	SetMotorParam	1	0	0	0	1	0	0	1
2	N/A	1	1	1	1	1	1	1	1
3	N/A	1	1	1	1	1	1	1	1
4	Irun & I hold	Irun(3:0)				Ihold(3:0)			
5	Vmax & Vmin	Vmax(3:0)				Vmin(3:0)			
6	Status	SecPos(10:8)			Shaft	Acc(3:0)			
7	SecurePos	SecPos(7:0)							
8	StepMode				AccShape	StepMode[1:0]			

Tabla 27: Instrucciones para SetMotorParam

SetOTPParam

Comando utilizado para reconfigurar la memoria OTP. Con este comando podemos modificar la dirección del esclavo.

OTPA direccionará los datos que queremos escribir al registro correspondiente.

El byte Pbit representa el patrón de bits que se desea programar. Por ejemplo, si se quiere programar los registros de la memoria OTP con los valores por defecto de Irun=0xD e Ihold=0x5, se debe ejecutar el comando SetOTPParam con los datos OTPA=0x03 y Pbit=0xD5.

SetOTPParam command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	SetOTPParam	1	0	0	1	0	0	0	0
2	N/A	1	1	1	1	1	1	1	1
3	N/A	1	1	1	1	1	1	1	1
4	OTP Address	1	1	1	1	1	OTPA(2:0)		
5	Pbit	Pbit(7:0)							

Tabla 28: Instrucciones para SetOTPParam

SetPosition

Comando utilizado para conducir el motor a una posición dada en relación a la posición cero, definida en un número de micro pasos.

SetPosition command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	SetPosition	1	0	0	0	1	0	1	1
2	N/A	1	1	1	1	1	1	1	1
3	N/A	1	1	1	1	1	1	1	1
4	Position byte1	TagPos(15:8)							
5	Position byte2	TagPos(7:0)							

Tabla 29: Instrucciones para SetPosition

SoftStop

Si se da este comando durante el movimiento del motor, provoca una inmediata deceleración a Vmin seguida de una parada sin importar si se ha alcanzado la posición final. Este comando puede darse en las siguientes circunstancias:

- La temperatura del dispositivo supera el umbral establecido
- Orden directa del microprocesador (master)

SoftStop command									
Byte	Content	Structure							
		bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Slave Address	1	1	OTP3	OTP2	OTP1	OTP0	HW	0
1	SoftStop	1	0	0	0	1	1	1	1

Tabla 30: Instrucciones para el SoftStop

Descripción del código

Como se ha comentado con anterioridad, el software que corresponde al controlador TMC222 sigue la misma estructura que el descrito para el sensor MLX90614. También contaremos con dos códigos, el principal y el que contiene todas las funciones que desarrollarán la comunicación.

Código principal

Al igual que en el caso del sensor, el código comenzará con los prototipos de las funciones que posteriormente detallaremos y con la implementación de los comandos descritos.

Las definiciones de los comandos que aparecen en la Figura 72 nos permitirán llamar a las funciones con los nombres de los comandos sin tener que recordar su valor hexadecimal.

```
#define GetFullStatus1 0x81;
#define GetFullStatus2 0xFC;
#define GetOTPPAram 0x82;
#define GotoSecurePosition 0x84;
#define HardStop 0x85;
#define ResetPosition 0x86;
#define ResetToDefault 0x87;
#define RunInit 0x88;
#define SetMotorParam 0x89;
#define SetOTPParam 0x90;
#define SetPosition 0x8B;
#define SoftStop 0x8F;
```

Figura 72: Definición de comandos para el código

Tras lo cual procederemos a declarar los buffer que contendrán los datos a enviar a las funciones. Estas funciones corresponderán con los comandos para implementar las instrucciones.

```
bufferSetMotorParam[0]= 0xC2;
bufferSetMotorParam[1]= 0x89;
bufferSetMotorParam[2]= 0xFF;
bufferSetMotorParam[3]= 0xFF;
bufferSetMotorParam[4]= //Irun & Ihold
bufferSetMotorParam[5]= //Vmax & Vmin
bufferSetMotorParam[6]= //SecPos,Shaft,Acc
bufferSetMotorParam[7]= // SecPos
bufferSetMotorParam[8]= //AccShape, StepMode
```

Figura 73: Buffer para la configuración del SetMotorParam

El primer byte corresponde a la dirección del esclavo siendo el resto bytes los contenedores de la información para la configuración del resto de parámetros.

```
bufferRunInit[0]= 0xC2;
bufferRunInit[1]= 0x88;
bufferRunInit[2]= 0xFF;
bufferRunInit[3]= 0xFF;
bufferRunInit[4]= //Vmax y Vmin
bufferRunInit[5]= //POS1 BYTE 1
bufferRunInit[6]= //POS1 BYTE2
bufferRunInit[7]= //POS2 BYTE 1
bufferRunInit[8]= //POS2 BYTE 1
```

Figura 74: buffer para la configuración del RunInit

Otro ejemplo de buffer que debemos implementar para el envío de datos es el de RunInit, de nuevo, la primera dirección es la del esclavo con la condición de escritura.

El resto de buffers son similares y de igual modo nos permitirán configurar los distintos parámetros del motor.

Las últimas instrucciones del código principal son las llamadas a las funciones.

Algunas funciones tan sólo necesitan la dirección del esclavo y el comando puesto que son instrucciones a realizar. Estas son las definidas en la Figura 75.

```
void ResetPosition (0x00, ResetPosition);
void GotoSecurePosition (0x00, GotoSecurePosition);
void HardStop(0x00, HardStop);
void ResetToDefault(0x00, ResetToDefaul);
void SoftStop (0x00, SoftStop);
```

Figura 75: Llamada a las funciones del programa

Sin embargo, el resto de funciones necesitarán, como se especificó para el caso del sensor, la dirección del esclavo, el comando y los buffer con la información de configuración, pero en estos casos los buffers contendrán también los dos primeros datos, la dirección del esclavo y el comando, por lo tanto sólo será necesario transmitir el buffer de datos (ver Figura 76).

```
void Init_I2C (unsigned char SCLL, unsigned char SCLH);
void SetMotorParam (unsigned char *bufferSetMotorParam);
void SetPosition (unsigned char *bufferSetPosition);
void RunInit (unsigned char *bufferRunInit);
```

Figura 76: Prototipos de las funciones del código del motor

```
void GotoSecurePosition (unsigned char S_Addr, unsigned char Command);
void ResetToDefault (unsigned char S_Addr, unsigned char Command);
void HardStop (unsigned char S_Addr, unsigned char Command);
void SoftStop (unsigned char S_Addr, unsigned char Command);
void ResetPosition (unsigned char S_Addr, unsigned char Command);
```

Figura 77: Prototipos de las funciones del código del motor sin buffer

Código funciones

En el segundo código del software del motor se implementan las instrucciones de las diferentes funciones antes mencionadas.

Hay una función por cada comando descrito y siguen el mismo modelo que las funciones del sensor, es decir, respetando el protocolo de comunicación del puerto serie I2C y basadas en la escritura y lectura de los registros de las memorias a través de buffers de información.

3.3 Descripción del hardware

El prototipo diseñado consiste en una placa que se conectará a la placa de pruebas para el microprocesador ARM7 llamada LPC2100, y al motor QSH4218.

La placa contiene elementos ya descritos en el primer capítulo de ésta memoria como son:

- El controlador TMC222
- El sensor de temperatura MLX90614
- La resistencia pull-up dinámica LTC1694

Y también contiene otros elementos no descritos aún y que se detallarán en este apartado.

Además, en ese mismo capítulo, el primero, se presentaron las características principales de:

- El motor paso a paso QSH4218
- El microprocesador ARM7 y su modelo LPC2129

Sin embargo queda describir la placa de pruebas MCB2100 que hemos mencionado.

3.3.1 Placa MCB2100

Placa de evaluación de Keil para el microprocesador LPC2129 que incluye. Estas placas están diseñadas para generar y probar programas de aplicación para los microprocesadores de Philips como el utilizado en nuestro prototipo. Contiene numerosos elementos para diferentes aplicaciones, como un altavoz, potenciómetros, Leds etc.

En nuestro prototipo utilizaremos las salidas SDA y SCL (Puertos 2 y 3) para el conexionado además de las fuentes de alimentación de 3’3V y las tierras de la placa.

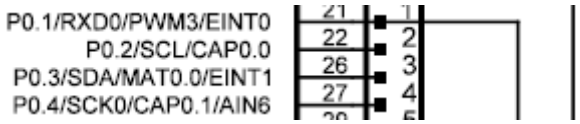


Figura 78: Detalle del esquemático de la placa MCB2100

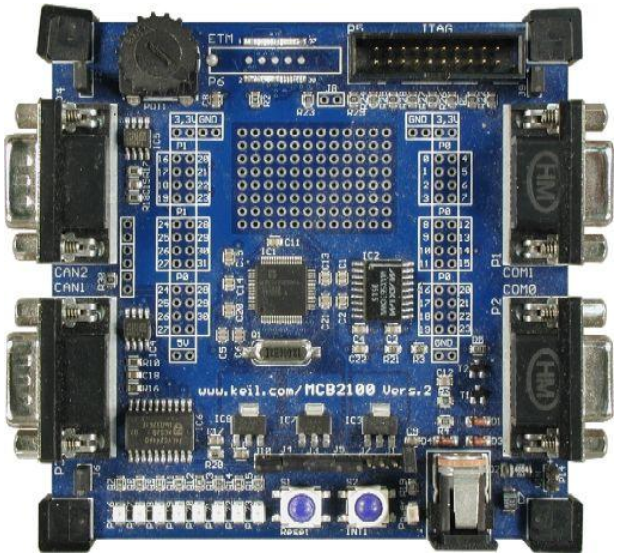


Figura 79: Imagen superior de la placa MCB2100

3.3.2 Hardware del prototipo

Para conseguir el objetivo propuesto en el proyecto hemos de conectar las diferentes partes del sistema que se comunicará mediante el protocolo I2C, para ello el esquema de bloques sería el mostrado en la Figura 80.

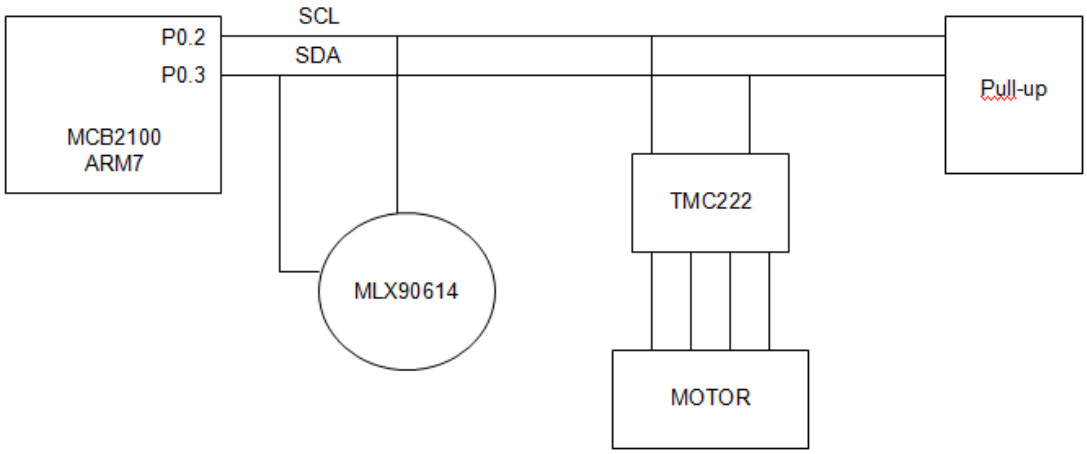


Figura 80: Esquema de bloques del sistema

En la Figura 80 se muestra un esquema básico y general del sistema de control de posición, sin embargo, hay elementos como el motor y el microprocesador que no se incluirán en la placa a crear sino que se conectarán a ella como elementos externos por lo tanto, el esquema mostrado en la Figura 81, se ha implementado con el programa Orcad y describe el conexionado del prototipo que hemos de implementar para nuestro proyecto.

La placa ha de medir aproximadamente 6x7 cm, es decir, 2350x2750 mils.

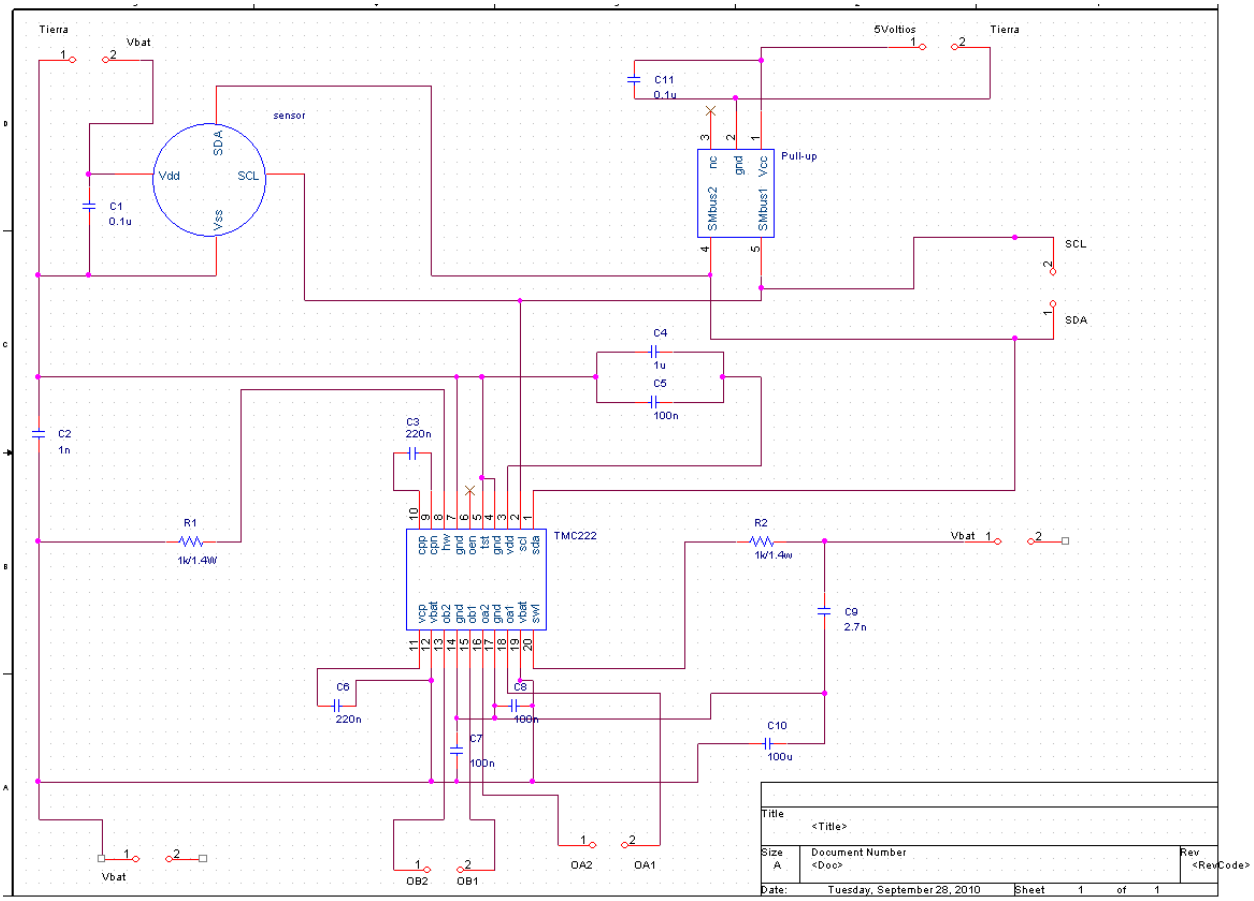


Figura 81: Esquemático en Orcad de las conexiones de la placa

En el esquema (Figura 81) se describen las conexiones que deben seguir los elementos para el correcto funcionamiento del dispositivo.

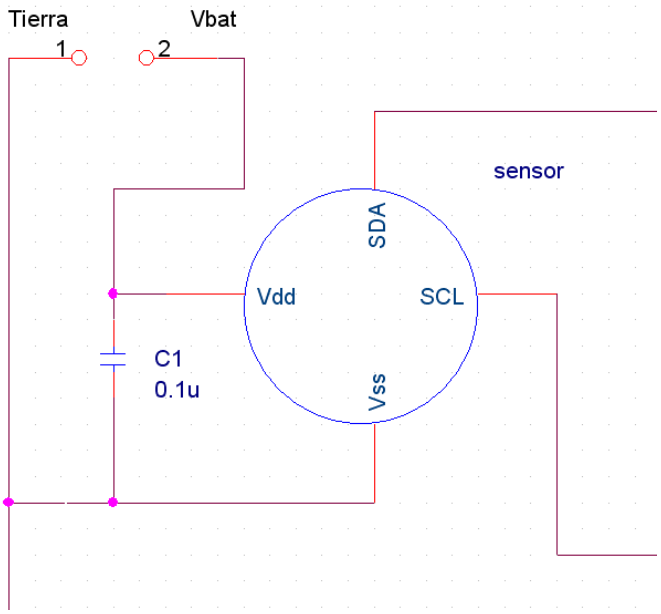


Figura 82: Detalle del conexionado del sensor

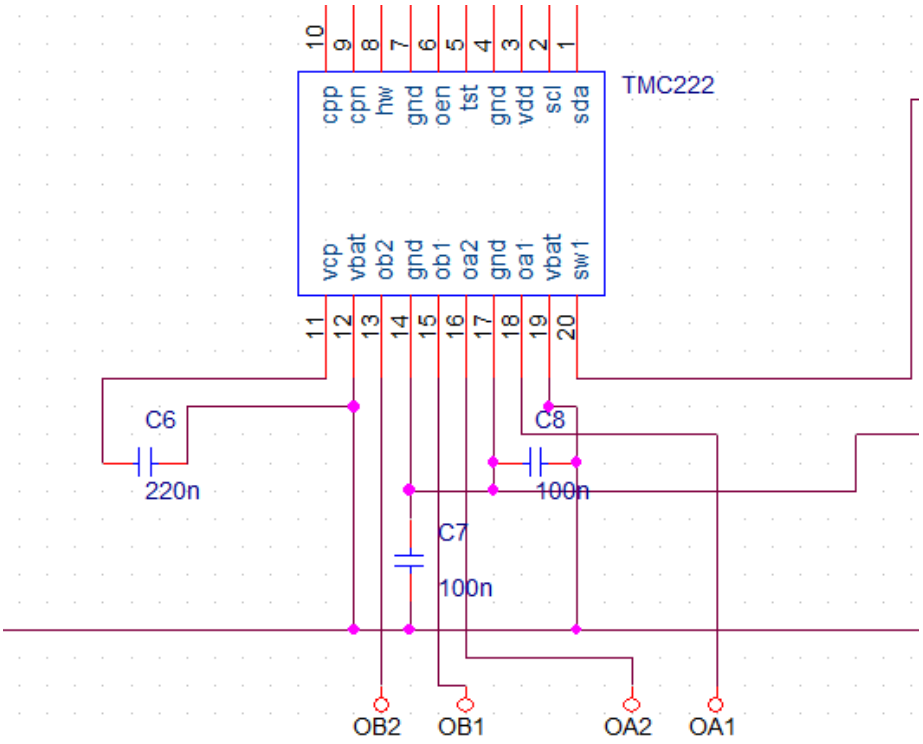


Figura 83: Detalle del conexionado del controlador

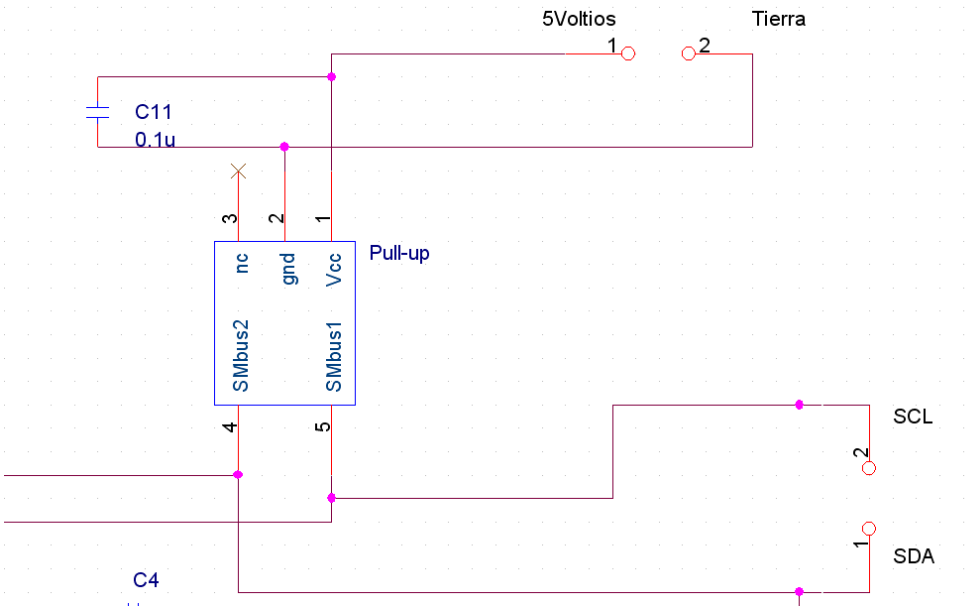


Figura 84: Detalle del conexionado de las resistencias pull-up

Para poder realizar este esquema y el posterior diseño de la placa impresa se han tenido que crear previamente las huellas correspondientes a los disposivos TMC222, MLX90616 y LTC 1694.

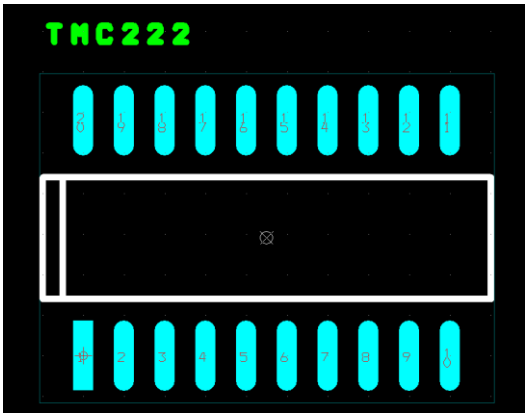


Figura 85: Huella para el controlador

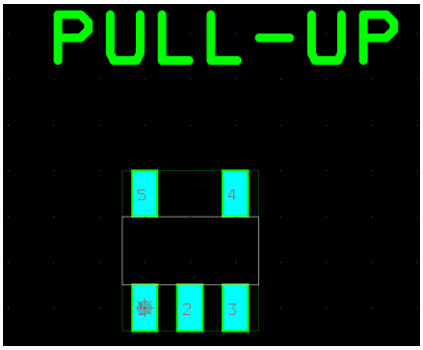
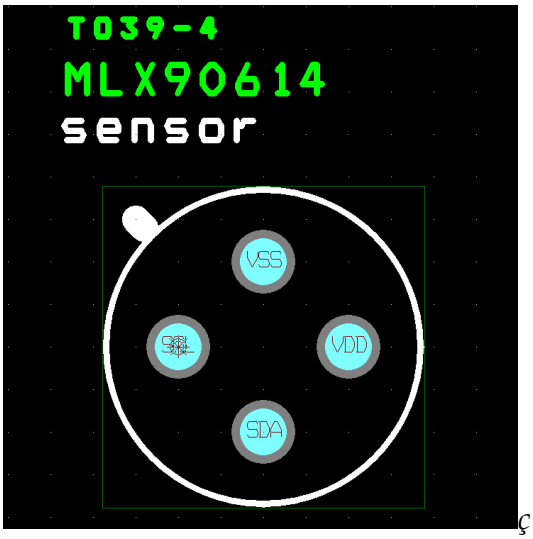


Figura 86: Huella para las resistencias dinámicas



ç

Figura 87: Huella para el sensor

En el esquema se puede comprobar que, además del sensor de temperatura MLX90614, el controlador TMC222 y la resistencia Pull-up, existen numerosos condensadores y otras unidades dispuestas en la placa. A continuación se detalla un listado de esos materiales con las referencias según la base de datos de RS amidata [web17]. Para todos los elementos descritos se han tenido en cuenta:

- Las medidas físicas de los dispositivos
- El tipo de montaje. Hemos elegido elementos en su mayoría de montaje superficial, no pasante, por su facilidad de soldado.
- El rango de temperatura en el cual pueden operar.
- Precio.
- Características específicas

Condensadores:

- De tántalo y valor 1uF, este condensador además debía tener un valor de ESR bajo. Código 684-4332

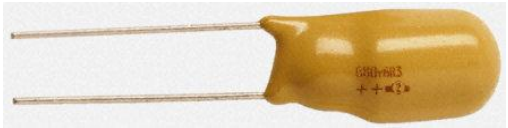


Figura 88: Condensador de Tántalo

- De 100nF necesitamos 5 unidades. Código 699-4948

Figura 89: Condensador de 100nF

- De 220nF necesitamos 2 unidades. Código 537-3915

Figura 90: Condensador de 220nF

- De 100uF, condensador polarizado con baja ESR.Código 538-2234

Figura 91: Condensadores polarizados

- Entre 2'7nF y 10nF, se ha escogido un modelo de plástico de valor 3'3nF, necesitaremos dos unidades de ellos. Código 622-4139

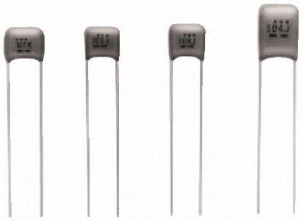


Figura 92: Condensadores de plástico

Resistencias

- Necesitaremos dos resistencias axiales de película de carbón y valor 1K y ¼ de watio. Código 135-847

Figura 93: Resistencia de carbón

Y esos son todos los elementos que forman la placa diseñada para el objetivo del proyecto. En la Figura 94 se muestra una imagen del circuito impreso rutado en las capas TOP y BOTTOM según el diseño anterior. Este esquema se ha desarrollado con la aplicación layout del Orcad.

Como se puede observar en la Figura 94, se ha dejado espacio en los bordes de la placa, que mide 2350x2750 mil, para el conexionado a través de Jumpers y conectores entre la placa diseñada y la placa MCB2100 así como para las conexiones con el motor a través de los terminales OB2, OB1, OA2 y OA1 de la Figura 94.

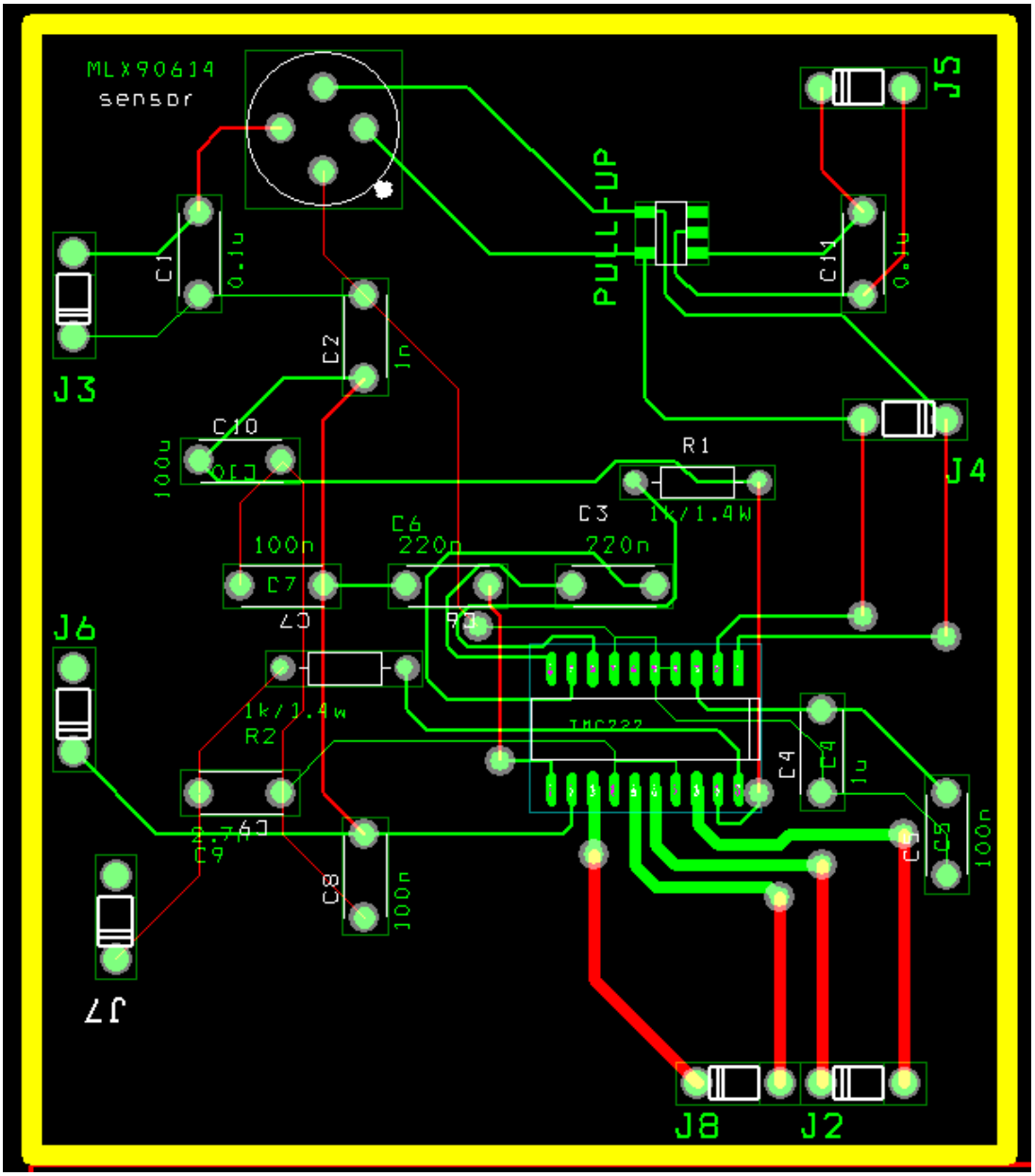


Figura 94 : Rutado del prototipo

Capítulo 4

Medidas y pruebas

Las pruebas y medidas realizadas en este proyecto han sido de software, es decir, la comprobación de la comunicación entre el microprocesador y el sensor a través del puerto I2C, la recepción y transmisión del bit de confirmación, la transferencia de los datos del buffer determinado, etc.

Algunas de las pruebas realizadas se muestran como capturas en las siguientes figuras.

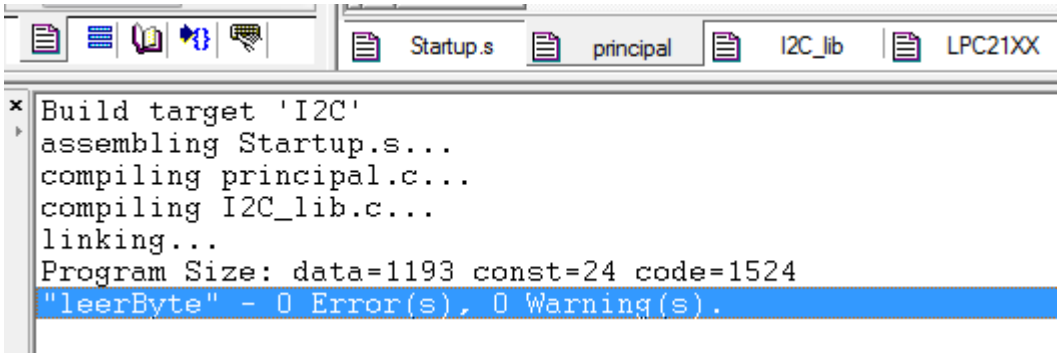


Figura 95: Captura de compilación

En la Figura 96 se pueden comprobar varios detalles como son, el nombre de las funciones implementadas (Startup.s, principal, I2C_lib) y también está abierta la pestaña con la librería LPC21xx. Además comprobamos que el código no produce errores de compilación.

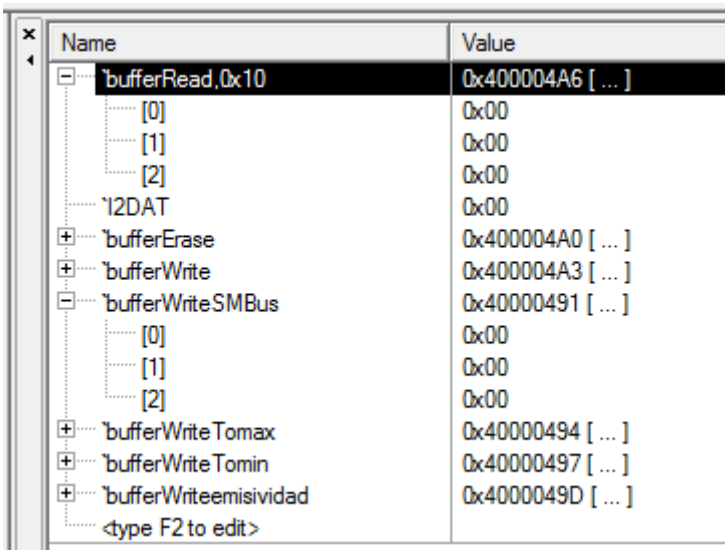


Figura 96: Ventana de variables a vigilar

En la Figura 96 podemos comprobar que antes de ejecutar el programa, los datos almacenados en los buffers son 0x00 y también que el registro I2DAT no contiene ninguna información.

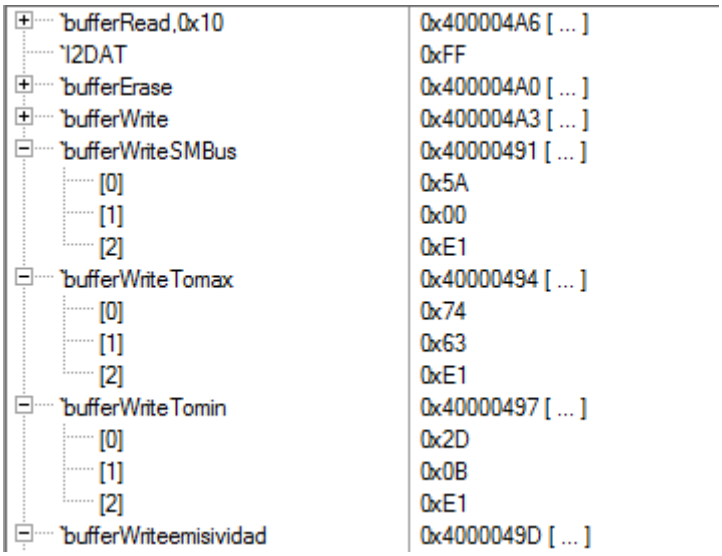


Figura 97: Ventana de variables a vigilar tras ejecutar el programa

Tras ejecutar el código comprobamos, gracias a la ventana de vigilancia de las variables elegidas, que los buffers han almacenado correctamente los datos que se les han proporcionado en el programa principal.

Sin embargo, esto no prueba nada acerca de la eficacia de la comunicación, para ello tenemos que visualizar la ventana del periférico I2C que inicialmente muestra un aspecto como indica en la Figura 98.

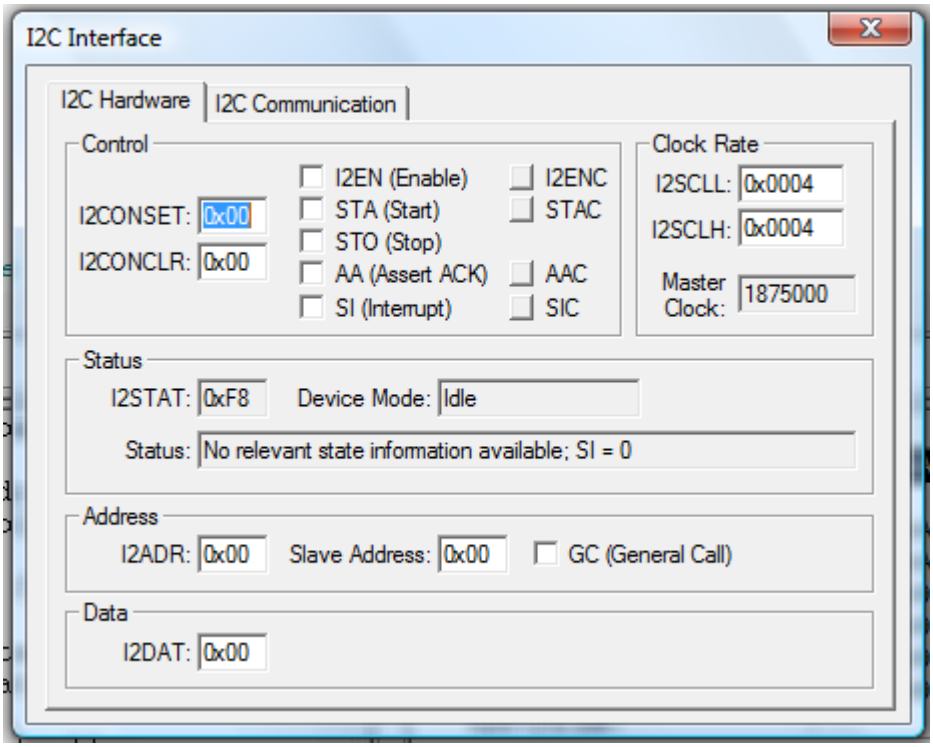


Figura 98: Ventana interfaz I2C inicialmente

Tras la ejecución del programa esta ventana presenta otro aspecto diferente (Ver Figura 99)

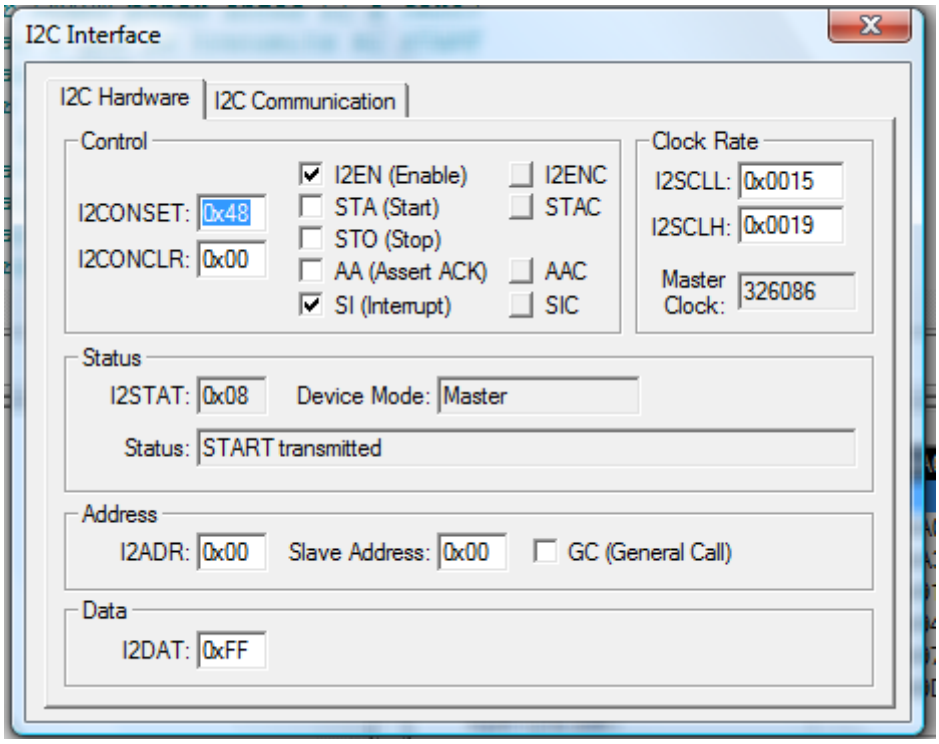


Figura 99: Ventana de simulación del I2C

En la Figura 99 podemos comprobar que se ha transmitido un START y que los registros I2CONSET e I2STAT han cambiado, además de que el registro I2DAT presenta un dato.

Pero para cerciorarnos de que la comunicación se ha llevado a cabo, no tenemos más que visualizar la siguiente pestaña (Ver Figura 100)

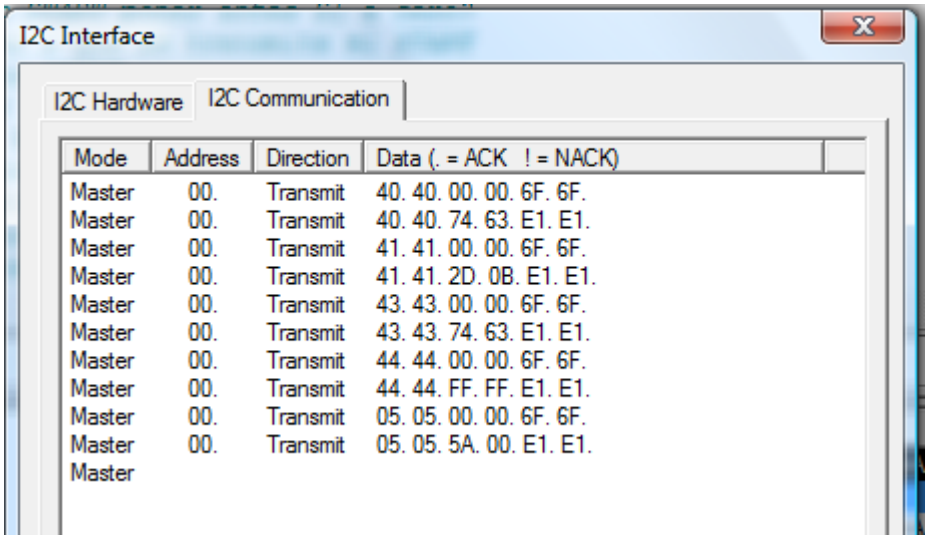


Figura 100: Vista del la ventana de comunicación I2C

O también, y gracias a la ejecución de una función exterior, también podemos observar la comunicación en la pantalla principal (Ver Figura 101)

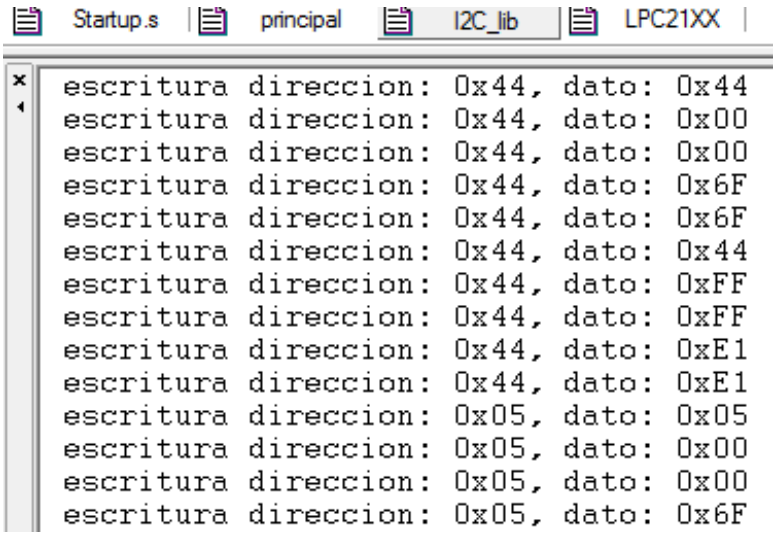


Figura 101: Detalle de la ventana principal de depuración

En las figuras podemos comprobar que aún no se ha cambiado la dirección del esclavo y que las acciones desarrolladas son de escritura de datos en la memoria EEPROM. Además en la Figura 101 visualizamos que se ha transmitido una condición de START y el programa se ha parado, esto se debe a que la siguiente ejecución es una instrucción de lectura de uno de los registros y en este caso debemos modificar nosotros el estado del bit ACK para que se de la comunicación, ya que al ser un programa de simulación, el sensor no está conectado y no hay esclavo que responda a la llamada, para eso se utiliza la interfaz I2C mostrada anteriormente. (Ver Figura 102)

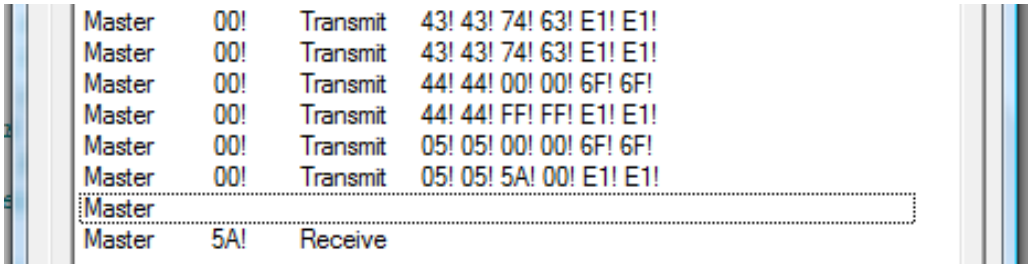


Figura 102: Detalle de la correcta recepción de los datos

Figura 103: Detalle del cambio de dirección del esclavo

No se han realizado pruebas con el software del motor debido a que se precisan datos de posicionamiento que hasta no tener una aplicación concreta no se pueden implementar, pero las funciones implementadas son exactamente las mismas, cambiando tan sólo los datos a transmitir.

Capítulo 5

Conclusión y presupuesto

5.1 Conclusión y líneas futuras

El desarrollo de un proyecto como este en el cual se comunican diferentes dispositivos con tecnologías distintas a través de un mismo bus de comunicaciones que tiene su propio protocolo, requiere un esfuerzo previo de estudio y comprensión de los elementos que lo conforman, de su conexionado, configuración y estructura interna.

Además se hizo necesario un estudio en profundidad del protocolo de comunicación serie I2C de Philips, puesto que es el método empleado tanto para la configuración de todos los dispositivos, como para la comunicación entre ellos, lo cual fue una dificultad añadida puesto que para cada envío/recepción había que implementar una serie de instrucciones de modo que los registros contaran con los datos adecuados.

Pero la mayor dificultad surgió del empleo del microprocesador ARM7 puesto que a pesar de ser de una tecnología parecida al 8051 que se utiliza en la carrera de electrónica, tiene numerosas peculiaridades nuevas que se debieron estudiar.

En general los resultados obtenidos con este proyecto han sido satisfactorios. Se ha conseguido el diseño de un dispositivo que sería capaz de controlar la posición alta-azimutal de una cámara y de medir la temperatura de manera que se pueda distinguir entre un objeto vivo o no.

Se desarrolló el modo de transmitir datos desde el microprocesador al sensor y al controlador del motor a través del puerto I2C y se comprobó su funcionamiento en caso del sensor con valores de prueba.

Además se realizó el conexionado de todos los sistemas y se diseñó la placa en la que deben ir soldados los componentes y conectados el motor y el microprocesador.

Cabe destacar que por falta de medios, presupuesto y tiempo, no se ha podido fabricar el circuito impreso, lo cual hubiese sido interesante sobre todo para realizar pruebas físicamente y no sólo con el simulador, pero el diseño está terminado tan sólo a falta de encontrar el presupuesto para fabricarlo.

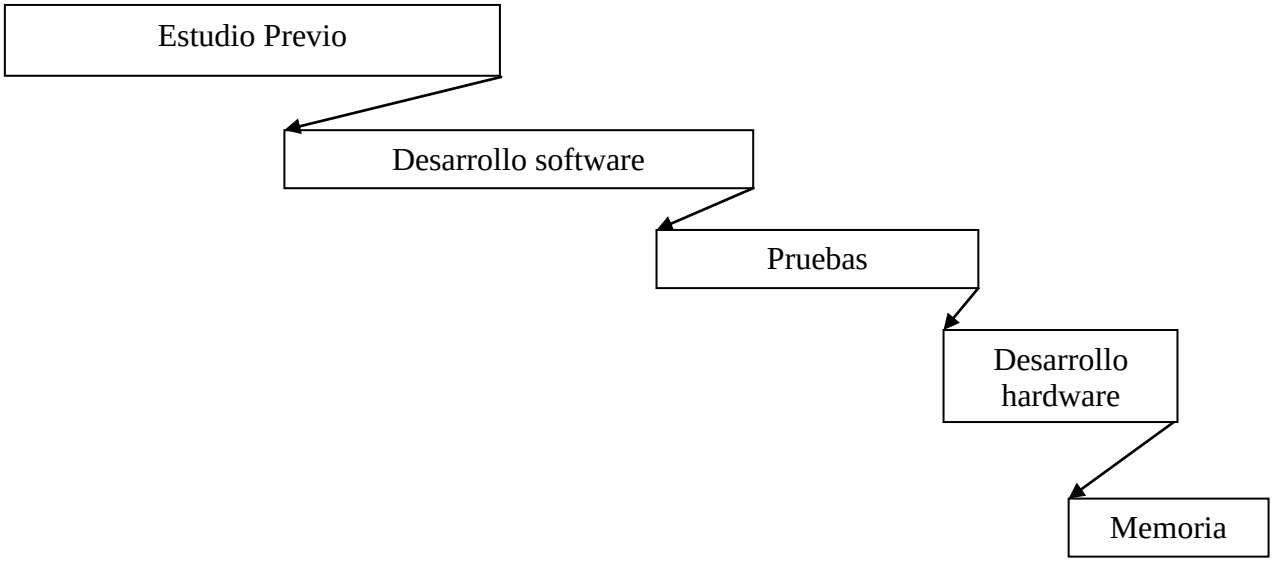
El software desarrollado permite la configuración del sensor y del controlador, sin embargo, no se ha podido dotar al mismo de medidas concretas puesto que dependiendo de la aplicación del sistema, el motor deberá presentar unas características diferentes tanto de velocidad, posición y demás parámetros que lo constituyen, por lo tanto, como líneas futuras de trabajo habría que realizar un software optimizado para alguna aplicación concreta en la que haya

que posicionar la cámara en una situación determinada y conociendo las temperaturas que se deben detectar.

En resumen, un futuro proyecto debería completar los datos a transmitir y enviar en el software de los dispositivos para realizar un control concreto de la posición y de la temperatura.

Otra línea futura de actuación sería la ampliación del proyecto con más de un motor. Habría que añadir otro controlador para este segundo motor. Esta línea de trabajo es una solución optimizada y muy sencilla de llevar a cabo puesto que el conexionado es similar al desarrollado en este proyecto y el software sería exactamente el mismo que para el otro controlador.

Diagrama de Gantt del proyecto



5.2 Presupuesto

Otro aspecto importante en la gestión de un proyecto es sin duda el manejo de los aspectos económicos del mismo, tales como la inversión en materiales o la contratación de trabajadores. En definitiva se trata de controlar el gasto final. Para la realización de esta labor, generalmente se utilizan herramientas específicas, pero dadas las características de este proyecto, con una estimación es suficiente.

El siguiente presupuesto se estructura en dos apartados uno referente a los materiales utilizados y otro referente al coste de personal.

Coste de los medios materiales

El coste de los materiales utilizados en este proyecto se detalla a continuación, asumimos que la empresa ya posee un ordenador y no es necesario adquirir otro. [web].

Además de los gastos en recursos humanos y el equipo, tanto hardware como software, durante todo el proyecto se utilizan materiales consumibles, como es el papel para fotocopias, herramientas de trabajo (soldador, etc.), cables de conexión, etc., por lo que al presupuesto hay que añadir una pequeña cantidad para gastos imprevistos, en nuestro caso, 20 euros serán más que suficientes.

Cantidad	Concepto	Fabricante	PVP.ud	Total
1	Motor QSH4218	Trinamic	33’80 €	33’80 €
1	Controlador TMC222	Trinamic	8’9 €	8’9 €
1	Sensor MLX90614	Melexis	15’88 €	15’88 €
1	Placa de test MCB2100	Keil	149 €	149 €
11	Condensadores	Varios		9’6 €
2	Resistencias	Tyco Electronics		0’036 €
1	Placa para cicuito impreso	Roth Electronic	5’75 €	5’75 €
		Base imponible		231’066 €
		IVA		16%
		Importe IVA		36’97 €
		TOTAL		**288’03** €

Coste del personal

Este proyecto puede ser desarrollado por un Ingeniero Técnico Industrial, el cual dará término al proyecto aproximadamente en dos meses, cuyo sueldo es 1650 €/mes. A esta cantidad habremos de sumarle el 32% correspondiente a la seguridad social.

Nº Trabajadores	Nº Meses	Tipo trabajador	Sueldo mensual	Total
1	2	Ingeniero Técnico	1650 €	3300 €
	2	Seguridad Social	32%	1056 €
		Coste de personal		**4356** €

Se ha asumido que todas las tareas del proyecto son llevadas a cabo por una sola persona, cualificada y capaz de realizar todas las fases del proyecto. En la práctica, son varias las personas que se verían involucradas en el proyecto, existiendo al menos una persona encargada del modelado y mapeado, otra para la programación, e incluso una tercera que se encargara de realizar las pruebas de control de calidad.

Coste total del proyecto

Concepto	Total
Coste de materiales	288'03 €
Coste del personal	4356 €
Imprevistos	20 €
Total	4664'03 €

Anexos

Bibliografía:

En este apartado se exponen tanto la bibliografía como los documentos electrónicos utilizados en la elaboración del proyecto y de la presente memoria.

Libros:

1. Paret, Dominique: "El bus I2C de la teoría a la práctica", Paraninfo, 2005
2. Vazquez, Juan; Lorenz G. Michael; García Nieto, Juan; Sanchez Reillo, Raúl: "Sistemas electrónicos digitales basados en microprocesador ARM7", los autores, 2009
3. Trevor, Martin: "The insider's guide to the ARM7 based microcontrollers: an engineer's introduction to the LPC00 series, Hitex, 2006
4. Recasens Bellver, María Auxiliadora: "Diseño de circuitos impresos con Orcad Capture y Layout V9.2", Thomson, 2002

Publicaciones electrónicas

1. Manual de usuario de LPC2119/2129/2194/2292/2294: http://www.keil.com/dd/docs/datashts/philips/user_manual_lpc2119_2129_2194_2292_2294.pdf
2. Manual de usuario UM10120 LPC213x. Capítulo 11, Interfaz I2C: http://www.nxp.com/documents/user_manual/UM10120.pdf
3. Manual de Entorno de Desarrollo del ARM. Departamento de Tecnología Electrónica. Judith Liu Jiménez, Almudena Lindoso, Marta Portela, Raúl Sánchez Reillo.
4. Notas de aplicación de comunicación SMBus con MLX90614, Melexis microelectronic integrated systems. [CD]

Datasheets

1. MLX90614, single and dual zone Infra Red thermometer in TO-39 [CD]
2. TMC222, Micro Stepping Stepper Motor Controler/Driver with Two Wire Serial Interface [CD]
3. QMOT motor QSH4218 manual [CD]

4. LTC1694, SmBus/I2C accelerator[CD]

5. Keil software MCB2140-v20 esquemático [CD]

6. The I2Cbus Specification[CD] : www.datasheetarchive.com/**I2C-bus**%20**specification**%203-datasheet.html

Referencias

[web1]: "Motores Paso a paso", Pilar García García:
http://www.sebyc.com/crr/descargas/motores_pap.pdf

[web2]: "Tutorial sobre motores paso a paso", todorobot:
http://www.todorobot.com.ar/informacion/tutorial%20stepper/stepper-tutorial.html

[web3]: "Motores PaP-Lección3", Luis Rueda:
http://perso.wanadoo.es/luis_ju/ebasica2/mpp_03.html

[web4]: "servos", Robotics.com:
http://perso.wanadoo.es/luis_ju/ebasica2/mpp_03.html

[web5]: "Transductores de temperatura": Desi-iteso:
http://www.desi.iteso.mx/elec/instru/cap4.pdf

[web6]:"Transductores de temperatura", Wikiciencia:
http://www.wikiciencia.org/electronica/electricidad/ttemperatura/index.php

[web7]: "Transductores de temperatura"Capitulo16, Dr. Juan José González de la Rosa. http://www2.uca.es/grup-invest/instrument_electro/ppjjgdr/Electronics_Instrum/Electronics_Instrum_Files/temas/T16_trans_temp.PDF

[web8]: "Transductores de temperatura", JMIndustrial:
http://www.jmi.com.mx/documento_literatura/Transductores_temperatura.pdf

[web9]: "Temperature sensors Pa-RTD", Rdf:
http://www.rdfcorp.com/anotes/pa-rtd/pa-rtd_04.shtml

[web10]: "Lecciones de electronica, resistors PTC",
http://www.ifent.org/lecciones/PTC/ptc.asp

[web11]:"medidor de temperatura con termopar", Jorge:
http://iindustrial.obolog.com/medidor-temperatura-termopar-86703

[web12]: "Principio de funcionamiento de los termómetros infrarojos", PCE:
http://www.pce-iberica.es/medidor-detalles-tecnicos/instrumento-de-temperatura/principio-funcionamiento-termometro.htm

[web13]: "Pull-up", wikipedia http://es.wikipedia.org/wiki/Pull-up

[web14]:"El microcontrolador",Reiner Torres Labrada
http://www.ucontrol.com.ar/wiki/index.php?title=El_microcontrolador

[web15]:"Diseño y construcción de un robot bípedo caminante", Jose Emmanuel Torres:
http://cybertesis.upc.edu.pe/upc/2009/torres_gj/html/sdx/torres_gj.html

[web16]:"Comunicación bus I2C, descripción y funcionamiento", Eduardo J. Carletti: http://axxon.com.ar/rob/Comunicacion_busI2C.htm

[web17]:"Pagina de compra de componentes on-line"
http://es.rs-online.com/web/

En el CD adjunto se encuentran lo códigos, datasheets y otros documentos utilizados en la elaboración del proyecto